ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA Y SISTEMAS DE TELECOMUNICACIÓN

PROYECTO FIN DE GRADO

TÍTULO: Desarrollo e Implementación de Espacios Verdes Inteligentes Basados en Tecnologías M2M

AUTOR: Jesús Manuel Belmar Ocaña

TITULACIÓN: Grado en Ingeniería Telemática

TUTOR (o Director en su caso): Javier Lucio Ruiz-Andino, Telefónica I+D

DEPARTAMENTO: Ingeniería y Arquitecturas Telemáticas

VºBº

Miembros del Tribunal Calificador:

PRESIDENTE: Rafael Herradón Díez

VOCAL: Rubén de Diego Martínez

SECRETARIO: Lourdes López Santidrián

Fecha de lectura:

Calificación:

El Secretario,

RESUMEN

En esta memoria se muestra el desarrollo del proyecto realizado para la creación de un sistema implantado en espacios verdes de las ciudades. A partir del estudio previo del mercado en torno a las *Smart Cities* se plantea el desarrollo de un sistema que mejore la sostenibilidad, la eficiencia y la seguridad de sistemas de riego, de medidores de la calidad del aire o estaciones meteorológicas instalados en los espacios verdes y jardines de las ciudades. Sirve de demostrador de cómo mejorar la eficiencia del consumo de agua y energía, además de añadir la funcionalidad de monitorizar la información recibida por sensores instalados en estos espacios en tiempo real a través del sistema desarrollado.

La finalidad de este proyecto es la de crear un demostrador que sirva de ejemplo e inspiración para proyectos futuros que se basen en las mismas herramientas que son ofrecidas por Telefónica.

El demostrador consiste en el control, gestión y mantenimiento de estos espacios, de modo que se pueda realizar un control y un mantenimiento de los dispositivos y sensores instalados en cada localización y la monitorización de la información suministrada por dichos elementos gracias a la plataforma M2M mediante los simuladores de sensores que ofrece Telefónica para demostraciones. Desde la aplicación, y mediante tecnologías REST y WEB, se realiza la comunicación con los elementos ofrecidos por Telefónica. La información de los sensores almacenada se muestra mediante la creación de informes y gráficas, mostrando la información recibida de los distintos parámetros de medida posibles, como son consumo de agua, temperatura, humedad y medidas de calidad del aire entre otras disponibles. Además, hay un espacio disponible para localizar y gestionar los dispositivos de los que se disponga existiendo la posibilidad de creación de alertas y avisos en función de lo que requiera el usuario.

La solución creada como demostrador tiene dos componentes realizados por el estudiante.

El primer componente desarrollado consiste en una plataforma web mediante la cual se pueda gestionar en tiempo real los dispositivos instalados en estos espacios verdes, su localización, realizar informes y gráficas, mostrando la información recibida de los sensores y la gestión de reglas para controlar valores límites impuestos por el usuario.

El segundo componente es un servicio REST al cual se accede desde la plataforma web creada en el anterior componente y realiza la comunicación con los servicios ofrecidos por Telefónica y la identificación del proveedor del servicio con Telefónica a través del uso de un certificado privado. Además, este componente lleva a cabo la gestión de usuarios, mediante la creación de un servicio de autenticación para limitar los recursos ofrecidos en el servicio REST a determinados usuarios.

ABSTRACT

In this memorandum the development of the project carried out for the creation of an implanted into green spaces in cities system is shown. From the previous study of the market around the Smart Cities it arise development of a system to improve the sustainability, efficiency and safety of irrigation systems, meter air quality and green spaces installed weather stations and gardens of cities. It serves as a demonstrator of how to improve the efficiency of water and energy consumption, as well as adding functionality to monitor the information received by sensors installed in these spaces in real time through the system developed.

The purpose of this project is to create a demonstrator to provide an example and an inspiration for future projects based on the same tools that are provided by Telefónica.

The demonstrator consists of control, management and maintenance of these spaces, so that it can carry out checks and maintenance of devices and sensors installed at each location and monitoring of the information provided by these elements through the M2M platform sensor simulators offered by Telefónica. From the application, and using REST and Web technologies, communication with the items provided by Telefónica is performed. Information stored sensors is shown by creating reports and graphs showing information received from the different parameters as possible, such as water consumption, temperature, humidity and air quality measures among others available. Addition there is an available space to locate and manage devices which becomes available with the possibility of creating alerting and warnings based on what the user requires.

The solution created as a demonstrator has two components made by the student.

The first component consists of a developed web platform through which to manage real-time devices installed in these green spaces, displaying the information received from the sensors and management rules to control limit values by the user.

The second component is a REST service which is accessible from the web platform created in the previous component and performs communication with the services provided by Telefónica and the identification of the service provider with Telefónica through the use of a private certificate. In addition, this component performs user management, by creating an authentication service to restrict the resources available on the REST service to certain users.

ÍNDICE

ÍNDICE DE FIGURAS

LISTA DE ACRÓNIMOS

M2M: Machine to Machine

I+D: Investigación y Desarrollo

DCA: Data Collection and Analysis

PaaS: Platform as a Service

API: Application Programming Interface

neoSDP: New Service Delivery Platform

REST: Representational State Transfer

SOAP: Simple Object Access Protocol

3G: 3rd Universal Mobile Telecommunications System

SIM: Subscriber Identity Module

HTML: HyperText Markup Language

CSS: Cascade Style Sheet

JSP: JavaScript Pages

JSPF: JavaScript Pages Fragment

ECMA: European Computer Manufacturers Association

ISO: International Organization for Standardization

AJAX: Asynchronous JavaScript and XML

HTTP: HyperText Transfer Protocol

URI: Uniform Resource Identifier

URL: Uniform Resource Locator

UML: Unified Modeling Language

WADL: Web Application Description Language

CORS: Cross-Origin Resource Sharing

1. INTRODUCCIÓN

El desarrollo de este proyecto surge como ejemplo de demostración de Telefónica I+D, empresa dentro del grupo Telefónica, para que los desarrolladores de terceros puedan conocer cómo mejorar su negocio y cuan sencillo es utilizar las herramientas que ofrece para el desarrollo del mismo.

La finalidad de este proyecto está en crear un demostrador que sirva de ejemplo e inspiración para proyectos futuros que se basen en las mismas herramientas que son ofrecidas por Telefónica. Para ello se ha realizado un estudio en detalle de cómo se encuentra el mercado de aplicaciones enfocadas en las *Smart Cities*, tanto creadas por la propia Telefónica en su desarrollo interno, como por terceros usando su propia infraestructura.

De este modo se ha buscado una posible aplicación dentro de las *Smart Cities*, que añada nuevas utilidades a la gestión de parques y jardines que se encuentran diseminados a lo largo de una ciudad. Se busca incentivar la mejora del uso de recursos que estos lugares hacen, y conocer el estado en que se encuentran los mismos. Para ello se integran sistemas de control de la calidad del aire y estaciones meteorológicas además de sistemas que controlen el gasto de agua en el riego. Gracias a la instalación de estos sensores inteligentes y a las herramientas desarrolladas por Telefónica se pueden crear aplicaciones basadas en M2M que permitan al administrador de estos sistemas y a cualquier usuario de estas aplicaciones la posibilidad, en función del tipo de usuario, de gestionar, administrar y consultar mediante una interfaz gráfica sencilla un histórico de los datos almacenados por estos sensores(como por ejemplo la temperatura o la calidad del aire en una estación concreta entre otras). En definitiva se busca mezclar dos conceptos, que son dos pilares fundamentales en las ciudades de hoy en día, que es el tener espacios verdes, generadores de aire limpio, y ofrecer información en tiempo real de la meteorología y la calidad de los gases permitiendo a cualquier usuario conocer el estado de la condiciones atmosféricas de la estación más cercana.

1.1. SOLUCIÓN CREADA Y ESTRUCTURA DE LA MEMORIA

Para el desarrollo de esta solución se han creado dos plataformas: M2MService y Dashboard Application, que se describirán más en detalle en el capítulo Descripción de la Solución Propuesta.

En la figura 1 se puede observar la arquitectura básica de la solución creada, la cual será analizada más en detalle en el apartado de Arquitectura de la sección Descripción de la Solución Propuesta.

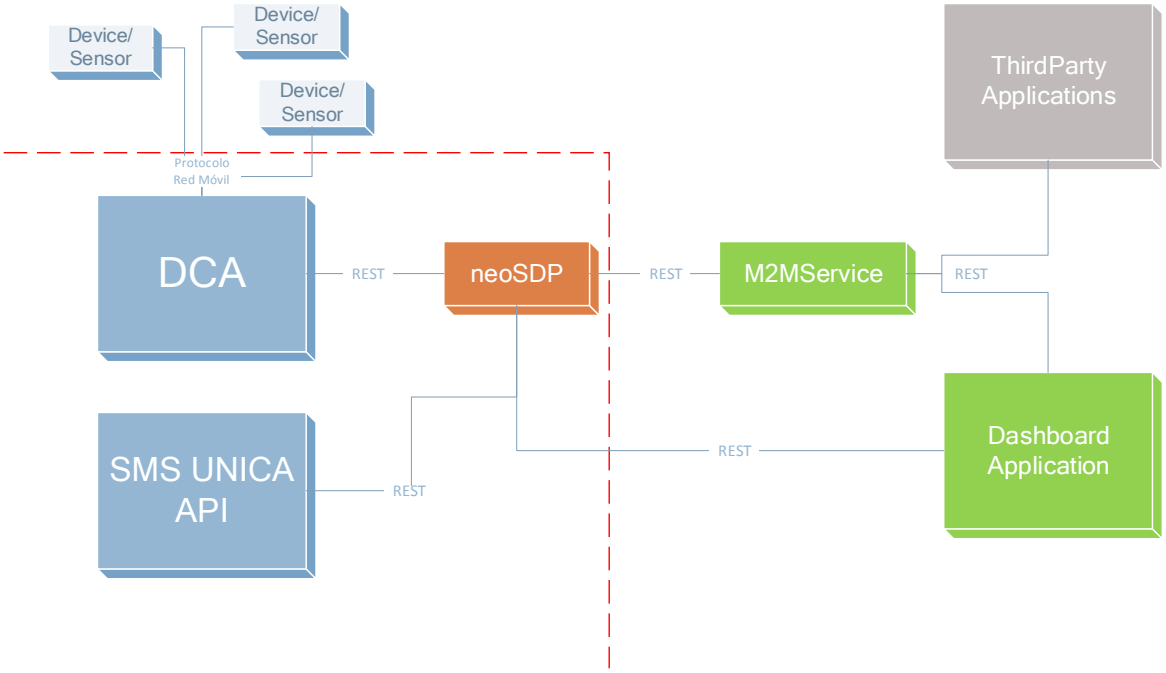

Figura 1. Arquitectura Básica de la Solución Propuesta

El primero de los componentes creados para la solución es el llamado M2MService. Esta plataforma ofrece la API de M2M creada por Telefónica. Las API de Telefónica son ofrecidas a terceros a través de la plataforma neoSDP [1]. Estas API están desarrolladas usando tecnología REST y SOAP. neoSDP añade una capa de seguridad en la comunicación, permitiendo solo el uso a terceras partes autorizadas previamente, de modo que exista un control de acceso restringido a las API ofrecidas. Esta capa de seguridad se ofrece mediante procedimientos de autenticación como OAuth, 2ways SSL y autenticación básica HTTP. La plataforma M2MService encapsula esta capa de seguridad ofrecida por neoSDP de manera transparente para el usuario final. Este servicio es exclusivo para cada usuario, ya que incorpora los credenciales necesarias que requiere neoSDP para su identificación como cliente fiable. M2MService incorpora un sistema de autenticación basado en Spring Security, permitiendo así únicamente la gestión de los dispositivos y sensores registrados en la plataforma DCA [2] a un administrador con los permisos delegados por parte del tercero que contrate este servicio.

Dentro del escenario de la demostración, la plataforma M2MService se ofrece como un posible servicio a contratar dentro de un Marketplace de servicios que tiene disponible Telefónica. El otro elemento del demostrador, llamado Dashboard Application, hará uso de este servicio, M2MService, para mostrar de manera gráfica los datos consultables a través del mencionado servicio. A partir de esta plataforma se demuestra como terceras partes pueden contratar los servicios exportados por Telefónica y de esta manera crear sus propias aplicaciones como por ejemplo, esta última aplicación nombrada. De esta manera se pretende demostrar cómo aplicar las capacidades que ofrece Telefónica a una idea de negocio dentro del marco de las *Smart Cities*, de manera sencilla y sin un

coste excesivo, ya que, gracias a la infraestructura que Telefónica tiene ya montada, solo es necesario una idea dentro de todas las posibilidades que ofrece las *Smart Cities* y sus propias vertientes, como por ejemplo son la *Smart Grid* o *Smart Mobility*.

La memoria se divide en varias secciones. Inicialmente se estudiarán los antecedentes, el estado actual del mercado y los proyectos y soluciones ya creadas que han servido de inspiración para la creación de este demostrador. A continuación se entrará en la parte más densa del documento, la descripción detallada de los componentes de la solución creada. En la parte de la descripción de la solución se explicará qué tecnologías y herramientas se han utilizado para dicho desarrollo y de qué manera han servido para llegar al resultado final. Además, se mostrarán los componentes, uno a uno, explicando cuál es su funcionalidad dentro del escenario creado. A continuación se mostrará un ejemplo de uso y las pruebas realizadas en la sección de Resultados donde a partir de simulaciones se crearán registros de medidas, comprobando cada utilidad creada y finalmente se comentarán las conclusiones a las que se ha llegado tras el desarrollo de este demostrador y las posibles futuras funcionalidades que se podrían añadir o hacia donde se podría enfocar esta solución, comentando las dificultades que han existido durante su desarrollo y las que podrían existir en desarrollos futuros en la sección final de Conclusiones.

1.2. TECNOLOGÍA M2M

M2M es el concepto que se ha otorgado a la comunicación inteligente entre distintos dispositivos. Esto significa que una vez que se ha procedido a desarrollar una inteligencia para el dispositivo, este se puede comunicar con iguales de manera autónoma.

De este modo, mediante esta tecnología se puede realizar una transferencia de información desde un dispositivo físico a otro, o a un dispositivo central que recoja la información recibida de un gran conjunto de dispositivos. Así pues se puede llevar a cabo un control y un mantenimiento de todo el conjunto de una manera centralizada, ya que desde el centro de la comunicación se puede conocer todo lo acontecido en cada uno de los elementos de la red formada mediante la tecnología M2M.

Durante los últimos años se han comenzado a desarrollar nuevos escenarios de negocio donde la informática y las telecomunicaciones forman parte importante dentro de éstos. Gracias a estos dos elementos se hace realidad el concepto de comunicación M2M donde se busca la interconexión entre dispositivos mediante la adquisición de una inteligencia artificial por parte de éstos. Los escenarios más conocidos, donde la tecnología M2M es una pieza fundamental en el desarrollo, son las *Smart Cities* y todas sus subáreas como son las *Smart Grids*, *Smart Mobility*, *Smart Meters* o *Smart Home*. De este modo, se aplica en sectores como la industria, agricultura, o en la electrónica del hogar entre otros.

Se considera que la tecnología M2M es la última evolución de Internet, pasando la barrera de la conectividad de los ordenadores y llegando a cualquier tipo de equipo electrónico que incorpore un microprocesador sea cual sea su contexto.

1.3. SMART CITY

La ciudad inteligente, mundialmente conocida como *Smart City* es un concepto que se ha ido desarrollando en los últimos años buscando una mejora en la calidad, sostenibilidad y eficiencia de las ciudades. No solo se centra en temas más escuchados mediáticamente como pueden ser la contaminación y eficiencia energética, sino también entran temas como puede ser la mejora del transporte o de los propios edificios que componen la ciudad.

El concepto de *Smart City* se puede descomponer en distintos temas y escenarios según el objetivo del estudio. En el caso de este proyecto, su desarrollo encaja dentro del escenario de *Smart Environment*. A continuación se muestran los escenarios más importantes dentro de la *Smart City*:

- Smart Mobility: Todo lo relacionado con el transporte dentro de la ciudad, desde los propios vehículos como toda la infraestructura que existe, carreteras, paneles informativos, semáforos, transporte público, etc.
- Smart Living: Dentro de este escenario se puede destacar conceptos como son el famoso Smart Home, el hogar inteligente. Además de todo lo que está relacionado con la mejora en la calidad de vida de las personas dentro de la ciudad de manera que mejore gracias a evoluciones tecnológicas.
- Smart Grids: En este escenario se puede encontrar todo lo relacionado con las evoluciones y nuevos conceptos en la distribución energética en las ciudades, de manera que se produzca de manera más eficiente sin costes innecesarios en su producción. De esta manera se están desarrollando dispositivos inteligentes utilizados para el control y mantenimiento de las redes eléctricas, llamados *Smart Meters*. En este documento se explicará cómo dispositivos formados por sensores y pequeños microprocesadores pueden desarrollar un desempeño similar, para el control en tiempo real y la comunicación bidireccional gracias a la tecnología M2M.
- Smart Government: Aquí se encuentra todo lo relacionado con los estudios realizados y desarrollos acerca de la manera de como la administración pública puede evolucionar y mejorar. Un caso ejemplo puede ser el gobierno de Reino Unido, que permite que a través de Internet se pueda realizar todo tipo de consulta y cualquier trámite con la administración. En España también se está trabajando sobre este escenario, implementando servicios telemáticos para todos los servicios y prestaciones ofrecidas por la administración pública, gracias a la creación del

DNI electrónico y el uso de certificados electrónicos para identificar de manera oficial a cada usuario. Servicios de este tipo por ejemplo es servicio electrónico para la declaración de la Renta, que se pueden realizar todos los trámites necesarios desde el hogar.

- Smart Economy: Se puede incluir aquí temas de I+D, las nuevas formas que se están estableciendo para la financiación de proyectos y empresas. Los proyectos de *crowd funding* donde colaboradores anónimos financian proyectos, sin tener realmente una participación activa en los proyectos desarrollados. Otro tema que se puede incluir en este escenario es la banca online, que de esta manera se puede realizar cualquier trámite con el banco sin tener que acudir a las oficinas. Y por último, otro tema que se puede destacar aquí, se encuentra en la evolución que es la forma que se tiene de revisar y recibir las facturas, actualmente con tendencia a poder revisarlas a través de Internet, y desapareciendo el formato en papel.
- Smart Environment: En este último escenario se encuentra todo lo centrado en la mejora y la reducción de la contaminación, todo lo que afecte al medio ambiente. Véase, sistemas inteligentes de riego, mejoras en la agricultura e incluso investigaciones para que los vehículos produzcan menor cantidad de gases que puedan afectar de manera negativa a la atmósfera. En el desarrollo de este proyecto, este escenario también ocupará parte importante, ya que en definitiva la aplicación que se desarrolla tiene como finalidad servir de herramienta para conocer gracias a los sensores inteligentes, en tiempo real, la situación de cada estación de manera que se pueda cambiar y mejorar los sistemas actuales de riego de manera eficiente y controlar en tiempo real el nivel de gases y fenómenos meteorológicos.

1.4. SMART SENSORS

Los dispositivos que son desarrollados para realizar una monitorización del medio son llamados sensores, sea cual sea el medio. Por ejemplo pueden ser dispositivos instalados como anteriormente se comenta, en el sistema eléctrico, de este modo realizar un control en tiempo real de la calidad y servicio de la red eléctrica. En nuestro caso particular, los sensores serán utilizados para medir valores del medio, como pueden ser a través de una estación meteorológica que disponga de medidores de humedad, presión, temperatura y de los niveles de calidad de los gases que contiene el aire o actuadores para controlar el flujo de agua y la cantidad de agua utilizada en el riego.

Otro uso común es evolucionar los actuales sensores que se utilizan en cualquier tipo de escenarios como pueden ser en la agricultura, o en los centros de estudio meteorológico, o en la medida de calidad de los gases de la atmósfera. Esta evolución consiste en añadir un microprocesador para que el sensor obtenga una pequeña inteligencia y así se permita una comunicación bidireccional con una central o con otros sensores, de

modo que gracias a la tecnología M2M se produce esta comunicación y se obtiene como resultado la capacidad de monitorización de la información de sensores en tiempo real y la capacidad para gestionar y administrar todos los dispositivos que se tengan asociados de una manera remota e instantánea.

Telefónica ha participado en grandes proyectos de I+D dentro del marco de la *Smart City*. Parte de esos proyecto han tenido como resultado el desarrollo de la herramienta que actualmente ofrece a terceros para que puedan implementar sistemas inteligentes mediante la tecnología M2M, el llamado DCA (*Data Collection and Analysis*). De este modo, mediante esta herramienta se ofrece la infraestructura necesaria para que solo sea necesario registrar el dispositivo en este DCA y gracias a que estos dispositivos llevan consigo sistemas de comunicación móvil, se puede monitorizar y gestionar en tiempo real de manera eficiente y sin un coste excesivo en la instalación de la infraestructura necesaria.

Como parte del desarrollo de la solución creada se explicarán las funcionalidades que este sistema ofrece y el servicio que tiene hacia el uso de *Smart Sensors*.

1.5. MARCO DE DESARROLLO EN TELEFÓNICA

Telefónica se ha embarcado en la investigación y desarrollo de nuevos sistemas y herramientas dentro del marco de las Smart Cities para ofrecer soluciones y nuevas funcionalidades a los sistemas actuales y de este modo obtener una mejora en la eficiencia energética, una gran mejora tecnológica y en definitiva, una mejora en la calidad de vida de las personas. Este desarrollo no solo ha sido por parte de Telefónica, sino que muchas empresas de telecomunicaciones e industriales se han puesto en marcha en la mejora de la calidad de vida a través de la mejora de los agentes que participan día a día en nuestra vida.

Como parte de proyectos desarrollados por Telefónica cabe destacar Smart Santander, un proyecto en el que participa con el objetivo de implementar servicios inteligentes que mejoren la movilidad de la ciudad además de la administración de la energía de la ciudad, teniendo sistemas luminosos controlados de manera inteligente, para la mejora de la eficiencia energética.

Otra gran pieza importante del desarrollo para la creación de Smart Cities es la herramienta que anteriormente se ha mencionado, el DCA, desarrollado en el área de M2M de Telefónica I+D. El DCA se ofrece como servicio PaaS y ofrece una infraestructura escalable que permite a terceros crear sus aplicaciones basadas en tecnología M2M, permitiendo así un rápido y sencillo enlace de comunicación entre los sensores y las aplicaciones creadas. Mediante el DCA se ofrece la capacidad de subscribirse a los eventos producidos por estos sensores, recibir notificaciones y alertas,

almacenar un histórico de todos los eventos creados por cada uno de los sensores que disponga la aplicación que quiera crear el desarrollador.

Para el desarrollo de este proyecto el DCA será pieza clave dentro de la estructura de la solución ya que como posteriormente se explicará detalladamente, consistirá en una demostración de las capacidades que ofrece Telefónica a partir de sus plataformas desarrolladas, siendo una de ellas este DCA.

2. ESTADO DEL ARTE

Durante los últimos años se han desarrollado las ideas de *Smart Cities* y M2M, además de todos los derivados de *Smart* relacionado con temas sobre la mejora del entorno de las ciudades, el hogar, la industria y el transporte. Como objetivos principales de estos proyectos están la mejora de la calidad de vida de las personas, la reducción de los efectos producidos en contra del medio ambiente y la salud de las personas, y la aplicación de cierta inteligencia a los sistemas ya instaurados.

Para el desarrollo de este proyecto se ha hecho un análisis de los proyectos que se están realizando en el entorno de Telefónica I+D, Telefónica y externos sin relación alguna con estas empresas. Se ha investigado y tomado nota de los puntos fuertes de cada proyecto además de sus puntos clave los cuales pueden ser punto de partida para nuevos proyectos de investigación para el desarrollo de nuevos sistemas y funcionalidades que se apliquen en los sistemas ya existentes.

A continuación se presentan distintos proyectos y soluciones creadas dentro del marco de las *Smart Cities* y la tecnología M2M que han servido de ejemplo y guía para el desarrollo del proyecto.

2.1 SMART CITY – TELEFÓNICA

Mediante el portal web de Telefónica especializado en la *Smart City* [3], Telefónica explica y fundamenta la necesidad de la evolución por parte de las ciudades. No solo aplicando una evolución en la parte física de las ciudades sino también en la administración de las ciudades. Para esta evolución de la ciudad se refiere a cómo aplicando las tecnologías de la información y las comunicaciones, las llamadas TIC, se debe de mejorar los sistemas actualmente establecidos para llevarlos a una mejora de la eficiencia de estos, que a su vez sean sostenibles, mejore la calidad de vida de los habitantes de las ciudades y que en definitiva mejore la gestión de los recursos disponibles.

De este modo aparece el concepto de *Smart City*, al utilizarse las TIC en el nuevo desarrollo de las ciudades, sus servicios públicos y componentes, para que sean más interactivos, eficientes y para que los ciudadanos puedan ser más participes de la ciudad y sus servicios.

Se comenta en este portal que el concepto de *Smart City* va de la mano del nuevo concepto IoT, Internet de las cosas, ya que en ambos conceptos se tiene como principales piezas claves las TIC y la tecnología M2M, que son utilizadas para establecer comunicaciones entre todo tipo de dispositivos, y que de este modo se establezca una red

de comunicaciones donde todos los elementos partícipes en la ciudad puedan estar conectados. Así adentrarse en el llamado Mundo Digital, y la esperada evolución de Internet.

Dentro del desarrollo de nuevos servicios para la mejora de los sistemas actuales de las ciudades se ofrecen diversos ejemplos en un gran número de campos distintos. Uno de ellos es la gestión inteligente de la movilidad en la ciudad, existiendo la posibilidad de distintas soluciones como son la gestión inteligente del transporte público. De este modo, por ejemplo con la implantación de dispositivos GPS, se puede llevar el control de las distancias y el tiempo estimado que queda para que un autobús llegue a su parada, así mediante aplicaciones móviles o servicios web saber cuándo hay que ir a la parada. Otros servicios son los servicios de parking, donde se controlan las plazas disponibles en una determinada zona para que de este modo se mejore el consumo de combustible que se puede desperdiciar en la búsqueda de una plaza libre además del tiempo gastado en ello. También mejoran la gestión de la movilidad en las ciudades la implantación de nuevos servicios de movilidad como son las bicicletas o el uso de transporte personal compartido. Otro ejemplo de desarrollo de sistemas que mejoran la movilidad son las aplicaciones para el control en tiempo real del tráfico en las ciudades para que de este modo se pueda elegir que recorrido se puede realizar si se quiere evitar los atascos, ejemplos de esto nombrados en el portal de Telefónica son MARTA (Movilidad y Automoción con Redes de Transporte Avanzadas) o el transporte de Barcelona [4], mediante su aplicación *tràns*it, donde se puede observar el tráfico de Barcelona en tiempo real.

Otro de los puntos importantes ya mencionados en el desarrollo de sistemas que permitan el control del estado del medioambiente en las ciudades, ya que se está teniendo mucha consideración para que el desarrollo de nuevos sistemas permita la mejora en la calidad del medioambiente haciendo que la contaminación de las ciudades sea menor. De este modo se están desarrollando sistemas para la monitorización en tiempo real del estado del medioambiente además permitiendo así la gestión remota de los dispositivos y sensores utilizados para dicha funcionalidad. Así, pueda mejorarse la eficiencia de la energía y de los recursos a partir del control del medioambiente creando sistemas autónomos e inteligentes.

Este último punto es importante para el desarrollo del proyecto que describe esta memoria ya que como finalidad al desarrollo es que los sistemas de riego de los espacios verdes de las ciudades puedan tener una gestión autónoma e inteligente a partir de los sensores y dispositivos instalados en cada uno de ellos.

2.2 SMART SANTANDER

Uno de los ejemplos de desarrollo de sistemas inteligentes en las ciudades es el proyecto europeo Smart Santander [5], financiado por la Unión Europea y donde participan empresas como por ejemplo, Telefónica, Ericsson o Alcatel-Lucent entre otras. En un proyecto desarrollado dentro del marco del 7º Programa Marco [6].

En el proyecto Smart Santander se han desarrollado sistemas inteligentes aplicados al transporte, al estacionamiento o al control del entorno. Ejemplos de estos sistemas son el sistema de avisos de parking instalado en el centro de la ciudad, que a través de una aplicación móvil puedes comprobar cuantas plazas disponibles existen en una zona determinada de la ciudad. También otro sistema implementado es el de transporte que como ya existe en otras ciudades se ha desarrollado un sistema de control de llegada de los autobuses a cada estacionamiento para poder saber el tiempo de espera para su llegada a la parada. Otro ejemplo, relacionado con la eficiencia energética, es el sistema de luminosidad que se ha implementado por otras zonas de la ciudad, donde en función del número de personas que se encuentren alrededor de las farolas en una determinada zona, variará la intensidad luminosa, para que el consumo de luz sea el más eficiente posible. Otro ejemplo, el cual sirve de base para la solución propuesta en esta memoria es el sistema de monitorización del entorno, de modo que a través de aplicaciones móviles puedes comprobar el estado de factores meteorológicos.

En este proyecto, Telefónica tiene una gran implicación, siendo parte importante en el desarrollo del proyecto. Como parte de este desarrollo presta el componente desarrollado por Telefónica, DCA, para el almacenamiento y control de los dispositivos instalados y los datos recogidos por cada uno de ellos. De este modo se demuestra las capacidades M2M que oferta Telefónica.

Este proyecto sirve de ejemplo en cuanto a que a través del desarrollo de aplicaciones móviles se puede realizar una monitorización de los factores meteorológicos y el control de la calidad del aire en un punto determinado de la ciudad de Santander.

2.3 PROYECTOS EN TELEFÓNICA I+D

En Telefónica I+D donde se ha desarrollado la plataforma de análisis y control DCA, se han desarrollado distintos proyectos a partir de esta plataforma, dando utilidad a las funcionalidades ofrecidas por este elemento.

Cabe destacar varios proyectos enfocados en la eficiencia energética del hogar. Estos proyectos están enmarcados en una rama de la *Smart City* llamada Smart Home, donde se enmarcan todos los proyectos relacionados con el desarrollo de nuevas funcionalidades para el hogar, unos cuantos ejemplos son la creación de sistemas

inteligentes para el control de la luz en la casa, el control remoto de los aparatos electrónicos del hogar, sistemas autónomos de calefacción, o sistemas de seguridad para cuando no haya nadie en la casa, entre otros ejemplos de la gran variedad existente, y que está por desarrollarse. En este caso, el enfoque tomado en el desarrollo de estos proyectos de Telefónica I+D están dirigidos a que el cliente de una compañía eléctrica pueda llevar el control y la monitorización en tiempo real del gasto energético de su hogar sin depender de terceros o externos.

Los proyectos están basados en un portal donde el cliente puede gestionar sus dispositivos, y en la instalación de dispositivos y sensores que se puedan conectar con el DCA y de este modo el cliente pueda llevar la gestión de estos. En el portal se puede realizar un control de los dispositivos instalados, ver el consumo actual de cada uno de ellos, realizar un informe del gasto energético de cada uno de ellos y además existe la posibilidad de activar o desactivar los conectores eléctricos a estos dispositivos, para que de este modo dejen de funcionar si se requiere. Los aparatos electrónicos, en su conector llevan instalados un conector inteligente que es el que realiza la comunicación con el DCA para transmitir la información a la base de datos y así pueda gestionarse el uso de cada aparato electrónico conectado en el hogar.

Estos proyectos sirven de demostración de las funcionalidades ofrecidas por Telefónica para que terceros puedan contratar estos servicios. Los proyectos aquí descritos son *Energy Manager* [7] y *PowerHouse* [8].

Estos proyectos han servido de ejemplo y guía para poder desarrollar las funcionalidades que se ofrecen en el portal desarrollado en la solución propuesta en esta memoria, ya que muestran claramente el partido que se le puede sacar a las funcionalidades ofrecidas por Telefónica para crear sistemas inteligentes enmarcados en las *Smart Cities* que hacen uso de la tecnología M2M.

2.4 OUTSMART

El proyecto OUTSMART [9] es un proyecto realizado dentro del programa de investigación y desarrollo *7th Framework Programme* de la Unión Europea y de *Future Internet Private Public Partnership*.

Este proyecto está enfocado en crear cinco ecosistemas innovadores distintos. Cada uno de ellos dirigido a la mejora de una de las funciones que se realizan en la ciudad y que pueden realizarse de una mejor manera y más eficiente. Contribuyendo así a que el suministro de servicios públicos y los recursos disponibles se realice de una manera más sostenible y que su efecto dañino en el medio ambiente sea menor que el actual.

Los cinco ecosistemas han sido planteados en cinco ciudades distintas, Santander, Berlín, Trento, Birmingham y Aarhus. Y estos ecosistemas son *Waste Management*, *Water*

and Sewage, *Sustainable Transport*, *Smart Lightings and Smart Meters* y *Water as a Resource*. *Smart Lightings and Smart Meters* complementa los desarrollos del proyecto Smart Santander mencionado anteriormente. El resto de escenarios creados están dirigidos a la mejora de distintos servicios de las ciudades, *Waste Management*, en Berlín, es la mejora del sistema de gestión de los residuos, monitorizando el nivel de capacidad de los depósitos de basuras de modo que se pueda gestionar de manera inteligente las rutas que realizan los camiones de recogida. *Water and Sewage*, en Aarhus, está dirigido a la mejora del sistema de alcantarillado y el control de las aguas residuales de la ciudad de modo que se controle de una manera más eficiente el gasto del agua en la ciudad y así poder reutilizarla ya que en esta ciudad las lluvias son abundantes y controlar la calidad del agua potable para los ciudadanos. *Sustainable Transport* es un ecosistema desarrollado para la ciudad de Birmingham, mejorando toda la infraestructura de transportes para la ciudad, desde los autobuses, trenes y metro hasta toda la señalización de la ciudad, además de sistemas de transporte sostenibles como son las bicicletas utilizadas para el transporte en el centro de la ciudad. Finalmente *Water as a Resource* desarrollado en Trento, es un sistema desarrollado para mejorar la red de comunicación desde las zonas rurales de las montañas con la ciudad de Trento mejorando así la distribución del agua en esta provincia.

Estos escenarios muestran diversos ejemplos de desarrollo de nuevos sistemas para mejorar el entorno de las ciudades, mejorándolas tecnológicamente y además mejorando también el entorno que les rodea, intentando que las emisiones y efectos sobre el medio ambiente sean menores. Han servido de ejemplo para la elección de un proyecto dentro del *Smart Environment* como finalidad de la solución propuesta, y además muestra como proyectos de este tipo pueden llegar a hacerse realidad en las ciudades apoyados por las instituciones como es la Unión Europea, y no siendo solo prototipos que no llegan a realizarse.

2.5 Desarrollo de un Modelo de Ciudad Inteligente que Asegura la Utilización Óptima de los Recursos Existentes en Ciudades de Todos los Tamaños

En este informe [10] se presenta un modelo de *Smart City* donde se muestra cada rama de desarrollo dentro ésta. A partir del estudio de distintos informes realizados por empresas como son IBM o Hitachi se muestran los conceptos más importantes y que destaca sobre todo que las nuevas funcionalidades y servicios que se pretenden crear para la *Smart City*, están enfocadas en igual medida tanto a la mejora en la experiencia de los habitantes de las ciudades como en la mejora en la eficiencia energética y en el efecto dañino sobre el medio ambiente. Además explica cuáles son los motivos por los que se plantea el cambio de las ciudades, motivos como son el cambio en el estilo de vida de las personas, la informatización de los servicios, los grandes movimientos demográficos hacia

las ciudades y la preocupación por el empeoramiento del medio ambiente por culpa de la contaminación.

A través de este informe queda muy claro cuáles son los puntos de actuación dentro de la ciudad para hacerla inteligente. Uno de esos puntos es el entorno, y a partir de esta cuestión se toma como finalidad del proyecto la mejora del entorno de modo que a través de la creación de sistemas inteligentes se pueda reducir el consumo y gasto energético y así reducir el efecto producido por las ciudades sobre el entorno. Cada proyecto de investigación desarrollado dentro del enfoque de la *Smart City* debe tener en cuenta de igual manera, como se ha comentado antes, el medio ambiente y la experiencia de usuario de los habitantes de las ciudades.

3. DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA

A continuación se va a describir la solución creada, comenzando con la definición de la misma, especificando en qué consiste el demostrador, cuáles son sus fundamentos, y cuál es su finalidad. Seguidamente, se mostrará cuáles son los objetivos marcados y planteados para que el desarrollo de la misma se haya llevado a cabo. Se definirán las tecnologías y herramientas utilizadas en el desarrollo, centrándose con mayor intensidad en las tecnologías utilizadas en el desarrollo de los componentes M2MService y Dashboard Application, que en definitiva es la parte más importante independientemente de la importancia superior que tiene el componente DCA, que sin su existencia esta solución no podría haberse llevado a cabo exitosamente. A continuación se planteará la arquitectura de la demostración realizada, visualizando cada componente que es partícipe en la solución. Inicialmente, se va a mostrar la estructura desde un punto de vista superficial hasta un punto de vista más en detalle, viendo primeramente todo el conjunto de componentes que la conforman, desde una visión más externa y menos técnica, y a continuación las funciones que tiene cada componente, especificando técnicamente en qué consisten. Una vez mostrada la estructura se llevará a cabo el desglose en detalle de cada componente, especificando la tecnología utilizada para su desarrollo y las funcionalidades que ofrece.

3.1 DEFINICIÓN

La solución se realiza como demostrador de las capacidades de telecomunicación que ofrece Telefónica a terceros. Se van a mostrar las utilidades que ofrece el DCA desarrollado por Telefónica I+D y como a partir de las herramientas que ofrece Telefónica se pueden desarrollar aplicaciones basadas en la tecnología M2M, de modo que gracias a estas herramientas se permite el intercambio de datos en tiempo real entre los dispositivos con capacidades de red móviles y las aplicaciones creadas. Todos estos datos intercambiados y el control de los dispositivos se gestionan gracias a toda la infraestructura que ofrece el compone DCA.

Como tema principal y motivación para la creación de este demostrador, se hace una búsqueda inicial de qué tipo de aplicaciones se han desarrollado internamente en Telefónica para demostrar las capacidades del DCA, y se concluye que el entorno de un escenario donde los jardines públicos forman parte del sistema, añadiéndoles la capacidad de monitorizar los fenómenos que se produzcan a su alrededor, y que de esta manera, se pueda llevar un control de la contaminación, la calidad del aire, fenómenos atmosféricos y el gasto de agua producido en el riego de los jardines podría ser una idea válida y valiosa de las capacidades que se quieren demostrar en el desarrollo de esta solución.

Por tanto, como finalidad de la solución, se obtiene el desarrollo e implementación de espacios verdes inteligentes basados en el uso de tecnologías M2M. La creación de un portal donde se pueda gestionar qué dispositivos se encuentran instalados en cada jardín, realizar de manera sencilla el registro y el borrado del registro de estos dispositivos en el sistema, la creación de alertas para establecer un control de los límites de riesgo que se puedan alcanzar. Además existe la posibilidad de la visualización del registro de medidas realizadas por cada dispositivo, ya sea gráfica o textualmente. De este modo cada usuario, ya sea administrador o un simple visitante, pueda observar y controlar en tiempo real los valores de la calidad del aire, temperatura y humedad entre otras.

Para demostrar las capacidades ofrecidas por Telefónica a terceros a través de la plataforma neoSDP, se opta por demostrar el uso y capacidades del DCA. De este modo, se desarrolla un nuevo elemento, llamado M2MService, que de una manera sencilla y cómoda añade los componentes necesarios para autenticarse de manera fiable contra la capa de seguridad de neoSDP. Además, añade una gestión de usuarios para el uso de este servicio, de modo que solo puedan utilizar las funcionalidades de modificar y borrado ofrecido por el servicio si eres un usuario administrador. Con esto se promueve la creación de servicios a partir de este M2MService, creados por terceros.

3.1.1 OBJETIVOS

El objetivo principal del desarrollo del proyecto es la creación de un demostrador que muestre el uso desde la visión de un tercero de las capacidades de telecomunicación que ofrece Telefónica a través de la plataforma neoSDP. Como se puede observar en la figura 2, se contextualiza la situación de los elementos a desarrollar dentro de un posible escenario real.

Para este demostrador se elige las capacidades que ofrece la plataforma DCA y el servicio de mensajería SMS, en menor medida, pero existen otras numerosas API de Telefónica ofrecidas a través de la plataforma neoSDP. El uso de la API de SMS se realizará mediante el envío de mensajes a terminales móviles, durante la gestión de las alertas producidas.

Otro objetivo además de demostrar las capacidades que ofrece el DCA, es también el mostrar que funcionalidades son ofrecidas para el caso particular de gestionar un jardín público, desde la monitorización de los datos recibidos por los dispositivos instalados en él hasta la posibilidad de gestionar de un modo autónomo e inteligente el control sobre los dispositivos de riego que se encuentren instalados. Esto se producirá en función de que debido a un gasto elevado, se alcance una determinada cantidad o se registre una precipitación, con lo que provoque que independientemente de la programación que exista sobre estos dispositivos se pueda alterar su funcionamiento por agentes externos.

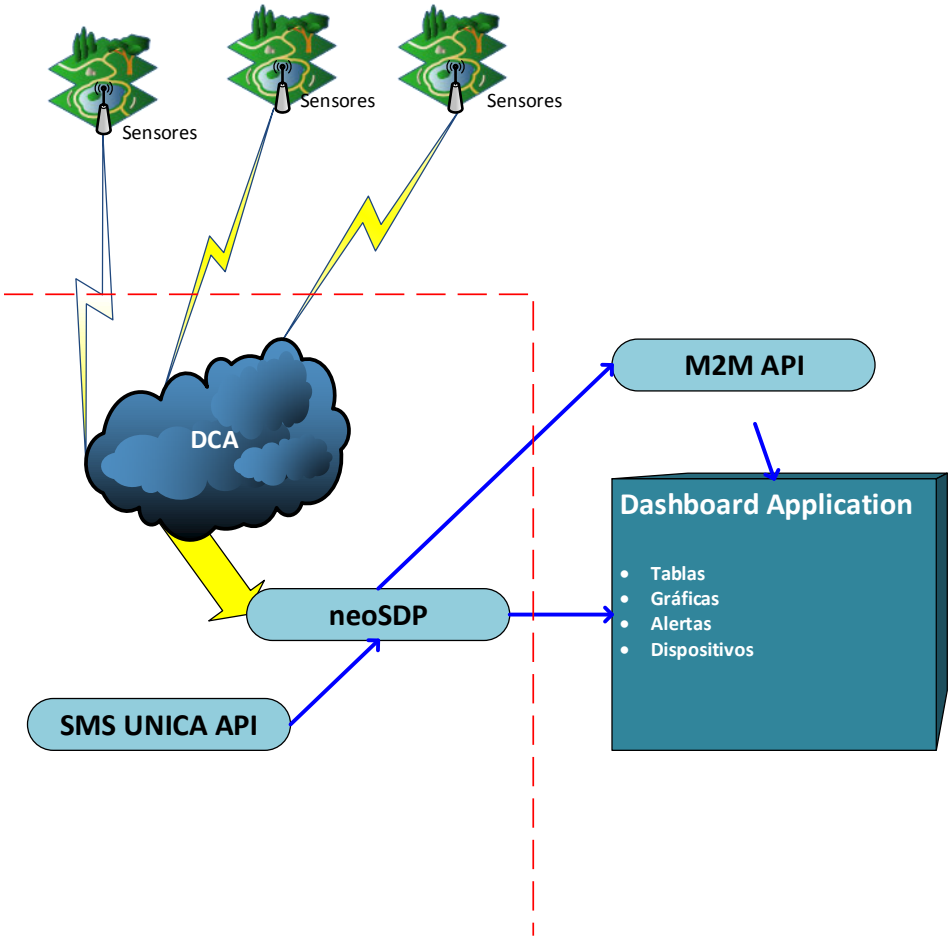

Figura 2. Escenario de la Solución

Como objetivo final del desarrollo de M2MService y Dashboard Application está el demostrar las múltiples posibilidades que existen de plantear ideas de negocio usando las capacidades ofertadas y más concretamente del DCA.

3.1.2 Herramientas Utilizadas

Este apartado está centrado en explicar y detallar brevemente qué herramientas software han sido utilizadas para el desarrollo de los componentes de la solución propuesta.

Eclipse

- Es una plataforma dirigida a la programación en distintos lenguajes de programación como pueden ser Java, C, C++, C#, JSP, entre otros.
- Desarrollada por Eclipse Foundation [11]. El proyecto original, "The Eclipse Project" fue originalmente creado por IBM en noviembre de 2001.

En 2004 se creó la Eclipse Foundation como fundación sin ánimo de lucro y comenzó a ser una plataforma de código abierto.
- *Software open-source*, dirigido a ser una plataforma abierta al desarrollo de nuevas funcionalidades para la plataforma.
- Posee una comunidad de desarrollo que ha ido ganando reputación en poco tiempo debido a las nuevas y útiles funcionalidades que se han ido desarrollando para la plataforma en forma de *plugins*.
- Ha sido utilizada en el desarrollo de todo el código fuente del proyecto.

APACHE TOMCAT
- Es un servidor web de *servlets* y JSP.
- Desarrollado por la Apache Software Foundation [12]. En 2009 Sun donó su desarrollo de su implementación de Java Servlets y Java Pages Specifications a Apache Software Foundation. La primera versión fue la 3.0 y actualmente está en fase beta la versión 8.
- Es utilizado para desplegar *servlets* creados en Java y que de esta manera puedan ser accedidos de un modo externo.
- En el caso de nuestra solución, ambos componentes, Dashboard Application y M2MServices son servicios desplegados en un Tomcat 6.0.

3.1.3 TECNOLOGÍAS

En este apartado se citan las tecnologías utilizadas en el desarrollo del proyecto.

HTML5
- HTML es el estándar para la programación de páginas web.
- HTML5 [13] es la última versión del estándar que actualmente se encuentra como borrador de la W3C. Su anterior versión es HTML 4.01 sacada en 1999 y HTML DOM nivel 2.
- Ha sido diseñada especialmente enfocada para la multiplataforma, debido al número de dispositivos con los que se consume contenido web, ya sea PC, móvil, tablet o la misma televisión. Además de para reducir el número de *plugins* a incluir en cada documento para poder ver y reproducir un contenido correctamente, ya sea multimedia o de cualquier otro tipo.
- Esta versión del lenguaje HTML ha sido desarrollado mediante una cooperación entre el consorcio W3C y el grupo WHATWG (*Web Hypertext Application Technology Working Group*).
- En esta versión se han desarrollado nuevos elementos con sus respectivas etiquetas de marcado. Dentro de las que se consideran más importantes se encuentran:

        - <canvas>: Para dibujos en 2D
        - <video>, <audio>: Elementos multimedia
        - <article>, <header>, <nav>, <section>, <footer>: Reestructuración de los elementos de contenido de la página, actualizando el valor del etiquetado a la actualidad, dejando el uso de la etiqueta para su uso inicialmente pensado.
        - Nuevos tipos de formularios actualizados, como puede ser calendarios, email, fecha hora, búsqueda, etc.
    - Con el desarrollo de HTML5 se busca renovar y mejorar la hoja de estilos utilizados en HTML, pasando a desarrollar de la misma forma CSS con su nueva versión CSS3.
    - Ha sido utilizado en el desarrollo de la interfaz web del componente Dashboard Application.

CSS3

- CSS [14] es el lenguaje de hojas de estilo utilizado para dar forma a un documento escrito de marcado véase los lenguajes basados en XML o como ejemplo más conocido, HTML.
- Ofrece un control total sobre el estilo y el formato que va a llevar el documento.
- Los estilos definen la forma de mostrar cada elemento HTML o XML.
- Mediante la hoja de estilo se crean reglas, aplicadas a cada elemento del documento. Cada regla puede afectar a uno o más elementos del documento. El formato de cada regla es el siguiente, un selector y una declaración:
    - h1 {color: red;}
- El selector es la parte mediante la que se enlaza con el elemento perteneciente al documento, mediante tres formas, sea indicando el tipo de elemento, el tipo de clase de un tipo de elemento, o a los elementos con un identificador concreto.
- La hoja de estilos puede acompañar en un documento independiente al documento HTML o XML o puede ir incluido en su cabecera.
- CSS3 ha supuesto una revolución en cuanto al diseño y creación de páginas web. Ha mejorado su código volviéndolo más simple y ha añadido nuevas opciones gráficas. El mayor inconveniente que se encuentra es que todavía no se muestra un 100% de compatibilidad con los navegadores web.
- En CSS3 se añaden nuevos tipos de selectores, de modo que se añade un mayor control sobre cada elemento del documento, inclusive su interacción con el usuario. De este modo se puede controlar sin necesidad de JavaScript o PHP la interactividad con el usuario como puede ser el seleccionar un

elemento con el ratón, marcar un texto, pasar sobre un elemento el ratón, entre otras nuevas funcionalidades.
- Durante el desarrollo de la solución propuesta CSS3 se ha utilizado ya que es la última versión de la tecnología CSS que comenzó en 1999.
- Ha servido para dar formato y estilo a la aplicación web Dashboard Application.

JAVASCRIPT

- [15]Es un lenguaje de programación utilizado para la creación de páginas web dinámicas. Una página web dinámica es aquella que contiene diferentes efectos gráficos, como pueden ser animaciones.
- Es un lenguaje de programación interpretado, es decir, que no necesita ser compilado.
- JavasScript es una marca registrada por Sun Microsystems. A pesar de su similitud en el nombre con el lenguaje de programación Java, no tiene ninguna relación.
- Se lanzó este lenguaje de programación en el año 1995 junto al navegador Netscape Navigator 2.0 después de la alianza entre Nestcape y Sun Microsystems. En el año 1997 se estandarizó el lenguaje Javascript mediante la especificación que Netscape mandó al organismo ECMA (*European Computer Manufactures Association*). El estándar fue denominado ECMA-262. La ISO adoptó este estándar, dando lugar al nuevo estándar ISO/IEC-16262.
- La sintaxis es muy similar a lenguajes de programación como son C o Java. Las reglas de escritura son bastante permisivas, ya que no es necesario la declaración de variables, o como sucede en C o Java, cada sentencia no es necesario que acabe con un ";".
- JavaScript actualmente sirve de fuente para la creación de nuevas librerías de nuevas funcionalidades aplicadas al desarrollo de entornos web, y herramientas como NodeJs, JQuery o JQPlot se nutren de este lenguaje para crear sus propias librerías de funciones para ser utilizadas para la mejora del entorno web.
- Ha sido utilizado en el desarrollo de la interfaz web del componente Dashboard Application.

JQUERY

- jQuery [16] es una librería basada en lenguaje JavaScript, creada en el año 2005, y lanzada en su primera versión estable en 2006.
- Es un software libre y de código abierto.
- Mediante esta librería se mejora en la interacción con los elementos HTML, ya que simplifica en gran manera el proceso.

- Al igual que el lenguaje base JavaScript, permite la agregación de animaciones y efectos al documento, además de añadir interacción con AJAX para realizar peticiones web.
- jQuery encapsula en funciones más simples y sencillas que de otra manera en simple código JavaScript necesitarían extensa cantidad de líneas de código.
- En el desarrollo de la aplicación web para la solución, esta herramienta ha sido muy útil ya que para crear una interfaz sencilla a la vez que útil ha sido necesaria por la comodidad y sencillez que ofrecía además de la gran comunidad que existe detrás mostrando ejemplos de uso.

JQPLOT

- jqPlot [17] es un *plugin* de jQuery para generar gráficas en una página web.
- Es un software libre y de código abierto.
- Mediante esta herramienta se permite de un modo sencillo crear todo tipo de gráficas y diagramas.
- Esta herramienta ha sido utilizada para la creación de los informes a partir de los datos recibidos de los sensores. Gracias a esta utilidad se ha permitido crear diagramas de línea y diagramas de barras con los datos recibidos de una manera sencilla y funcional.

JSP

- JavaServer Pages [18] es el nombre con el que se identifican las siglas JSP.
- Mediante esta tecnología se permite crear páginas web dinámicas basadas en HTML o XML.
- Para poder ejecutar código utilizando este lenguaje es necesario que sea ejecutado en un servidor web que permita el contener aplicaciones y *servlets*, ya que actúa como un *servlet*.
- Usa el lenguaje de programación Java, en su funcionamiento es similar a cualquier fichero programado en Java, necesita ser previamente compilado y esto se produce en el contenedor de *servlets*, en el caso de la solución propuesta se utiliza Apache Tomcat, como se ha mencionado anteriormente.
- Las principales ventajas que ofrece JSP son que aporta la posibilidad de crear clases, manejar y acceder a variables y datos creados de una manera minuciosa, y que al estar basado en Java, se puede compilar en múltiples plataformas sin tener que realizar cambio alguno sobre el código.
- En cuanto al desarrollo web, permite organizar de manera cómoda el contenido ofrecido en una web, de modo que se puede administrar cada componente de una manera autónoma y sencilla.
- Cuando se ejecuta el código JSP en la web se devuelve como salida código HTML y no queda huella del uso de JSP para su creación.

- Ha sido utilizado en el desarrollo del componente Dashboard Application y su integración entre web y Java.

JAVA

- Java [19] fue desarrollado por James Gosling Sun Microsystems en el año 1995.
- Es un lenguaje de programación orientado a objetos y basado en clases.
- Actualmente es uno de los lenguajes más utilizados en el mundo de la informática.
- Está orientado para que un mismo programa pueda ser ejecutado en múltiples plataformas sin necesidad de revisión del programa, como por ejemplo la ejecución de un programa en distintos sistemas operativos. Esto significa que además de la posibilidad de ejecutarse en distintos S.O., también se permita en plataformas sobre distinto hardware.
- Se creó también con la finalidad de que pudiera ser ejecutado de manera remota cliente-servidor, y gracias a interfaces y utilidades como RMI (*Remoted Method Indication*) o CORBA (*Common Object Request Broker Architecture*).
- En las plataformas de escritorio se ejecutan los programas sobre la JVM (Java Virtual Machine). Para poder ser ejecutado en cualquier plataforma, ésta debe de integrar el JRE (Java Runtime Environment).
- Se han definido tres tipos de Java en función de las capacidades de hardware y software y el tipo de usuario:
  - J2ME (Java Platform Micro Edition), orientada a entornos de recursos limitados, como podían ser los antiguos teléfonos móviles.
  - J2SE (Java Platform Standar Edition), para plataformas de gama media, donde se encuentra el usuario habitual de PC.
  - J2EE (Java Platform Enterprise Edition) orientada a entornos empresariales o Internet.
- En 2009 Oracle adquirió la compañía Sun Microsystems, pasando de esta manera Java a ser propiedad de Oracle.
- El desarrollo de la parte lógica del proyecto se ha realizado utilizando este lenguaje de programación.

SPRING SECURITY

- Spring Security [20] es un *framework* enfocado en la provisión de autenticación y autorización en aplicaciones desarrolladas en Java.
- Gracias a esta herramienta se puede solucionar problemas y vulnerabilidades en la seguridad de aplicaciones Java.
- Está enfocada a controlar la seguridad en entornos web, aplicándose en gran medida sobre la autenticación sobre API, seguridad en las peticiones HTTP, y seguridad sobre capas y dominios. También permite el control de usuarios

y la autorización adaptada de cada uno en función del tipo de usuario que sea.
- Mediante esta herramienta se pueden desarrollar implementaciones de seguridad más complejas como puede ser OAuth.
- En la solución propuesta en este proyecto, se ha utilizado Spring Security para desarrollar permisos a un administrador en el acceso a la API Rest creada en M2MService de modo, que solo pueda ser un administrador quien pueda modificar y crear nuevos dispositivos, permitiendo al resto de usuarios solamente al acceso a los datos registrados y datos acerca de cada dispositivo.

SPRING WEB MVC
- [21]Es un *framework* orientado a la web para la creación modelo-vista-controlador.
- En la actualidad se encuentra como el *framework* más utilizado para la creación de este modelo de negocio.
- Esta herramienta te permite tener separada la lógica de un elemento en un controlador y su interfaz en la vista, de modo que se encuentran independientes aunque a la vez están conectados ya que en la creación de la vista se pueden pasar los llamados objetos entre controlador y vista.
- Su librería de anotaciones para Java hace más sencillo la creación de servicios REST o cualquier otro servicio Web.
- Para la solución propuesta se ha utilizado tanto en el componente Dashboard Application como en el M2MService, ya que gracias a este *framework* su desarrollo ha sido menos complejo.

REST
- *Representational State Transfer* [22]. El nombre se toma de la tesis doctoral desarrollada por Roy Fielding en el año 2000, siendo uno de los autores principales en las especificaciones HTTP.
- Es una técnica de arquitectura software para sistemas hipermedias distribuidos.
- Mediante REST se identifica y manipula los recursos almacenados en un servidor web. De este modo para el acceso y manipulación a los recursos se debe realizar mediante una petición HTTP a una URI determinada para cada recurso.
- Los mensajes que se intercambian en cada petición HTTP son autodescriptivos, de modo que se conoce el tipo de operación a realizar en función de la cabecera del mensaje, por ejemplo las cabeceras HTTP que se pueden añadir son GET, POST, PUT, DELETE u OPTIONS. Además la sintaxis de la descripción sigue un estándar y estructura.

- La información intercambiada en el mensaje puede ser codificado en XML o en JSON.
- En el servicio desarrollado en la solución, M2MService, se ha utilizado esta tecnología REST utilizando como formato del mensaje el lenguaje JSON.

JSON

- JavaScript Object Notation [23]. Es un lenguaje de notación muy ligero utilizado para el intercambio de datos.
- Inicialmente utilizado en la notación literal de objetos en JavaScript.
- Debido a su simplicidad ya que no requiere el uso de cabeceras para indicar las librerías donde se encuentran detalladas las etiquetas posibles a utilizar, ha aumentado en gran medida su uso, y la sustitución del uso de XML. No siempre se hace una sustitución completa, sino que en función del servicio a desarrollar es necesario la utilización de ambas tecnologías, JSON y XML.

FASTERXML JACKSON JSON

- [24]Es un *framework* de Java para la transformación de objetos Java en entidades JSON de modo que, de una manera sencilla y rápida, se pueda transformar un objeto de Java en un simple texto en JSON.
- Ha sido de gran utilidad esta herramienta ya que ha permitido el manejo de entidades de JSON desde Java, convirtiéndolos en objetos Java y posteriormente su transformación para poder ser enviados como simple texto al servidor.

MAVEN

- Apache Maven [25] es una herramienta para la gestión de proyectos Java creada por Jason van Zyl de Sonatype en 2002. Actualmente pertenece a Apache Software Foundation.
- Es una herramienta que funciona de repositorio de librerías en red, de modo que con conocer la referencia a una librería, ésta se incluirá en tu proyecto Java.
- Utiliza un POM (*Project Object Model*) donde se describe el proyecto a construir y las dependencias a las librerías externas que necesite. Permite la adición y la eliminación de librerías al proyecto de manera muy dinámica.
- Se ha convertido en uno de los componentes indispensables en la creación de proyectos Java en el entorno profesional.
- Ha sido utilizado como repositorio de las librerías utilizadas en la herramienta de programación Eclipse.

En el apartado siguiente se establecerá visualmente como se sitúa cada componente dentro de la solución propuesta, ya sean los componentes desarrollados para la solución o los ya existentes.

3.2 ARQUITECTURA

Para mostrar la arquitectura de los agentes que componen la solución se partirá desde una visión más externa y básica hasta una visión más específica de cada componente de modo que se vea claramente de qué manera se ha desarrollado y cuál es su función dentro de la solución desarrollada.

3.2.1 ARQUITECTURA BÁSICA

En este apartado se puede ver una descripción de los componentes de la solución, ya sean componentes internos de Telefónica o componentes creados para la solución, los dispositivos y sensores que formarían parte del sistema e incluyendo la posibilidad de crear aplicaciones a terceros, como se quiere demostrar con este proyecto.

En la figura 1, mostrada anteriormente, se muestra de un modo sencillo todos los componentes que forman parte y el modo en el que se establece la comunicación entre sí.

En este caso, marcados con color verde, se reflejan los elementos desarrollados específicamente para la demostración, siendo el elemento llamado M2MService el servicio a contratar por terceros y que, de un modo sencillo, pueden exportar sus funcionalidades, creando, en el caso del demostrador, una aplicación web llamada Dashboard Application.

De color gris aparece el módulo llamado *ThirdParty Applications*, que indica la posibilidad que existe de crear aplicaciones que usen el sistema M2M creado por Telefónica mediante el uso de esta solución.

En color naranja se marca el componente llamado neoSDP, que sirve de enlace entre las API que ofrece Telefónica y los terceros, de modo que es quien se encarga de validar los accesos a las API de modo confiable.

La línea roja marca los componentes que son propiedad de Telefónica y han sido desarrollados internamente.

En blanco, aparecen los sensores y dispositivos que serán monitorizados en tiempo real gracias a la plataforma DCA que establecerá comunicaciones con estos sensores mediante las redes móviles como pueden ser: la red de comunicaciones móviles de alta velocidad, 3G.

Y finalmente, en color azul, se refleja el componente DCA, que es la pieza fundamental de la solución ya que en definitiva, lo que se quiere demostrar son las capacidades de escalabilidad, flexibilidad y multiservicio que ofrece el DCA.

El DCA ofrece al exterior una API REST con la que, de un modo sencillo, se permite gestionar los datos asociados a cada cliente. De este modo, esta API es ofrecida a través de la plataforma neoSDP que, como se ha comentado anteriormente, añade una capa

de seguridad para registrar al proveedor del servicio que contrata las funcionalidades ofrecidas por el DCA.

El componente desarrollado llamado M2MService se nutre de la API ofrecida por el DCA para añadirle la funcionalidad del control de usuarios y el certificado que se debe de comprobar en neoSDP para validar al proveedor de servicio contratante de los servicios ofrecidos por Telefónica. Una vez que el proveedor de servicio, el llamado "*Third Party*", contrata el servicio se le provee este M2MService junto con un certificado válido. A partir de este componente y el certificado, el cliente podrá crear sus propias aplicaciones M2M usando este componente, de modo que así pueda desarrollar sus propias ideas de negocio en el marco de la tecnología M2M.

Como ejemplo de la utilización de este componente, el M2MService, se ha desarrollado el Dashboard Application, que usa la API REST creada con el componente. Esta aplicación web es un demostrador de las capacidades ofrecidas por Telefónica, en este caso de la API de M2M que se ofrece.

A la misma altura queda el componente de color gris, *ThirdParty Applications*, que se refiere a cualquier aplicación desarrollada por terceros a usando el componente M2MService.

Los componentes llamado *device/sensor* son los sensores y dispositivos que mediante protocolos de redes móviles se comunicarán periódicamente con el DCA registrando los valores medidos por estos dispositivos. Cada dispositivo tendrá una tarjeta SIM para poder establecer comunicación.

Por otra parte, se encuentra el componente de la API de SMS ofrecida por Telefónica, SMS UNICA API. Este componente es el utilizado para el envío de SMS durante el demostrador. La lógica de funcionamiento es la misma que con el componente DCA, la API es ofrecida a través de la plataforma neoSDP sirviendo ésta de enlace entre las API de Telefónica y terceros. El uso de este servicio también es utilizado como demostrador de las capacidades ofrecidas por Telefónica al igual que el uso de las capacidades de M2M ofrecidas.

3.2.2 ARQUITECTURA DCA

El DCA [2] es un componente desarrollado por Telefónica. Es una plataforma horizontal M2M. Esta plataforma permite el desarrollo de aplicaciones basadas en la red de sensores de modo que a partir de distintas tecnologías se pueda desarrollar distintas aplicaciones enmarcadas en un entorno de M2M.

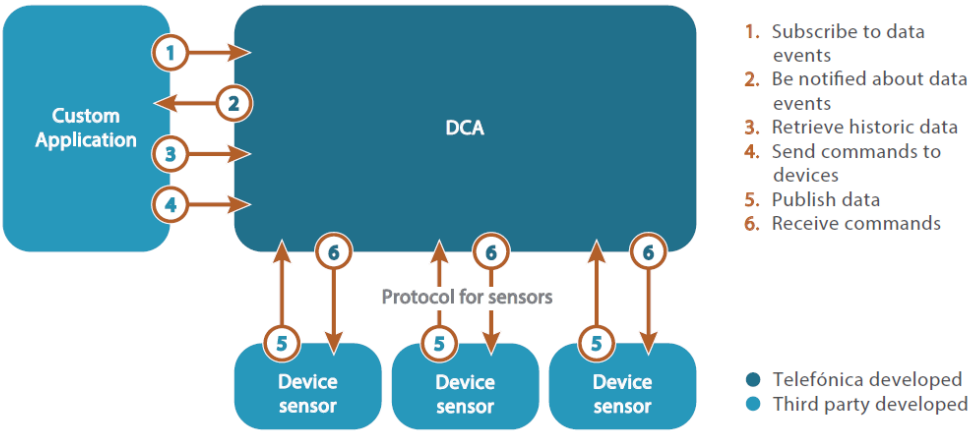

Figura 3. Arquitectura DCA

Ofrece una plataforma avanzada de inteligencia de datos para integrar nuevos servicios M2M de una manera fácil y rápida. El desarrollo se ha producido siguiente una serie de requisitos:

- Plataforma modular, genérica, flexible y multiservicio.
- Plataforma horizontal. Se centra en implementar diversas funcionalidades dentro del marco M2M.
- Modelado de Datos. Permite la publicación, subscripción y notificación.
- Repositorio de datos. Recoge la información de los sensores recibida a través de la nube ofreciendo una alta capacidad y disponibilidad, ofreciendo redundancia de HW/SW.
- Uso de API basadas en tecnologías web para el acceso a los servicios de la plataforma, como son REST y SOAP.
- Ofrece mecanismos de seguridad para el acceso.

El DCA se compone de tres elementos funcionales:

- *Device Gateway*: Esta es la capa que permite la interacción de la plataforma con los dispositivos registrados en el DCA. Su misión principal es reducir la complejidad de la red de sensores en cuanto a términos de protocolos y acceso a la red.

- *Real Time Event Processing Module*: Es la capa intermedia, el "*Middleware*" que recibe la información mandada desde un sensor, en formato de Sensor ML, el estándar utilizado y la almacena en función de los prerrequisitos y configuración previa asociada al consumidor.
- *Application Services*: Esta capa es la más externa, y su funcionalidad es ofrecer todos los servicios de las capas inferiores de un modo easy-to-use usando tecnologías web para ofrecer estos servicios. Este elemento funcional es con el que las aplicaciones que desarrollen servicios M2M, se deben de comunicar e interactuar.

3.2.3 ARQUITECTURA NEOSDP

Mediante la plataforma neoSDP se pretende ofrecer un acceso seguro y controlado a los clientes que contraten y quieran usar API desarrolladas por los *enablers* de Telefónica. En la figura 4 se muestra la arquitectura de la plataforma neoSDP.

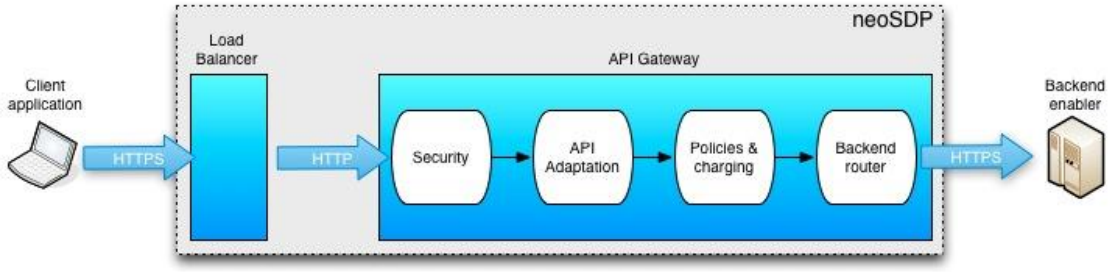

Figura 4. Arquitectura de Alto nivel de neoSDP

A partir de la figura anterior se pueden observar las distintas funcionalidades de cada componente. Primero con *Client application*, se entiende a cualquier aplicación desarrollada que haga uso de una API ofrecida por Telefónica. *Backend enabler* es quien ha desarrollado la API y la ofrece.

neoSDP se compone de:

- *Load Balancer*: Balanceador de carga que se encarga de redirigir las peticiones recibidas hacia los servidores que se encuentren con mayor disponibilidad en cada momento. Un ejemplo sencillo de cómo funciona este elemento se puede ver en la figura 5 mostrada en el tutorial de F5 [26].

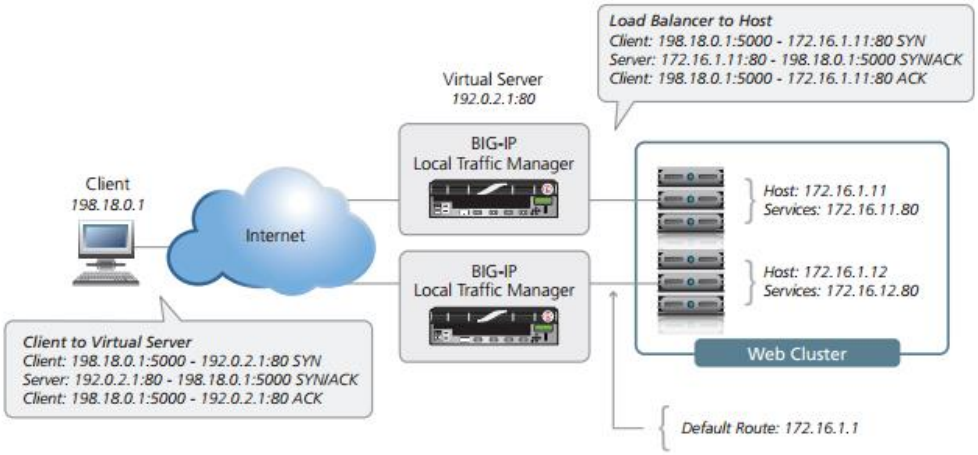

Figura 5. Balanceo de Carga Básica

- API *Gateway*: Adaptador intermedio entre el cliente y el *backend enabler* que ofrece la API. Se compone de diferentes procesos, que se pueden enumerar en los siguientes: proceso de seguridad, que añade autenticación, adaptación de API, para los diferentes tipos de representaciones y versiones, aplicación de políticas en función del consumo y carga de las API, y la carga de credenciales para poder acceder a las API ofrecidas por el *backend enabler*.

3.3 DIAGRAMAS

A continuación se va a proceder a mostrar diversos diagramas UML, para que se pueda analizar de una forma más visual y secuencial los procedimientos que se realizan en la solución propuesta. A su vez mostrará de una manera efectiva y simple las funcionalidades que ofrece dicha solución desarrollada.

En este documento, se mostrarán los diagramas más importantes y los cuales se han seleccionado para que se entiendan eficazmente las funcionalidades exportadas para el usuario de la solución.

3.3.1 DIAGRAMA DE CASO DE USO

El diagrama de caso de uso muestra el comportamiento que tiene un elemento hacia el exterior. De este modo se puede observar que funcionalidades ofrece, en este caso a un usuario o a un administrador del sistema.

Como se puede observar en la figura 6, el comportamiento hacia el usuario y hacia el administrador es distinto, ya que el administrador tiene posibilidad de gestionar por

completo el servicio desarrollado. Sin embargo, los usuarios solo tienen posibilidad de ver los datos recogidos por los dispositivos, generar informes a partir de estos datos y localizar geoposicionalmente dónde se encuentra cada dispositivo.

Existe esta división de derechos ya que el servicio desarrollado está enfocado a que un tercero, una vez que haya desarrollado su aplicación M2M pueda generar negocio a partir de ella y pueda ofrecer un servicio a sus clientes. De este modo los clientes se identifican con el usuario en el diagrama de caso de uso. Y por su parte, el administrador será el personal encargado de, como su nombre indica, la administración y gestión del servicio en su totalidad. En la siguiente figura se puede observar detalladamente el caso de uso de Dashboard Application.

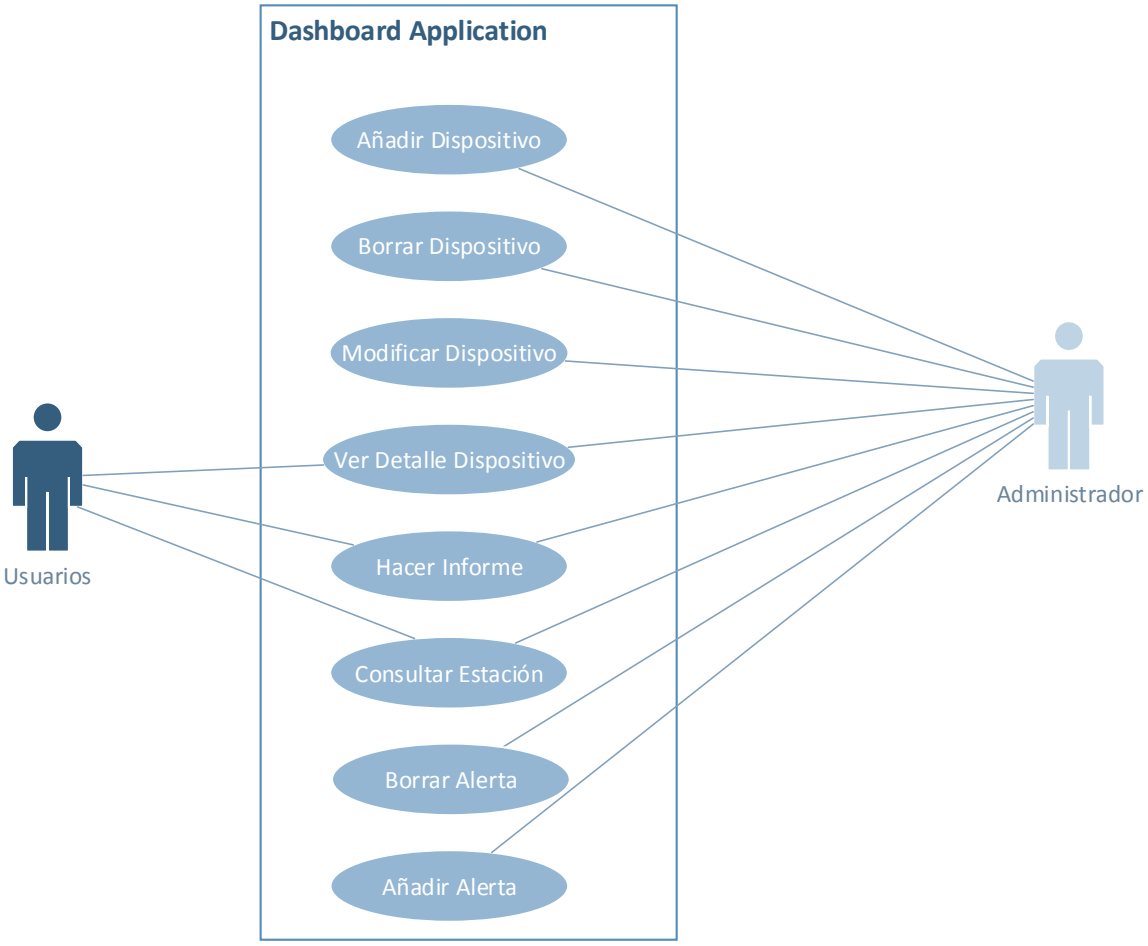

Figura 6. Caso de Uso

Las funcionalidades que ofrece el componente Dashboard Application, son las que principalmente se han desarrollado en toda la solución propuesta, y son las que visualmente se ofrecen en el Dashboard. Este componente hace de ventana hacia lo que se ha desarrollado en componentes internos como es, el creado específicamente para esta solución, M2MService, que aplica una capa de seguridad para usuarios y de esta manera proteger las funcionalidades contra los usuarios que no son administradores, como se ve en

el diagrama, y más internamente componentes como son la plataforma neoSDP y DCA, núcleo de la tecnología M2M que ofrece Telefónica.

3.3.2 Diagramas de Secuencia

A continuación se muestran distintos diagramas de secuencia donde se describe el uso funcional de la solución creada, de manera que se pueda observar como entran en juego los distintos componentes utilizados y cómo se comportan respecto a distintas acciones comenzadas por un usuario o un administrador. Se muestran los diagramas de secuencia de las acciones más características y de las cuales, se puede ver cómo sería la secuencia para otras acciones similares dentro de la solución.

Secuencia De Autenticación

En la figura 7 se puede observar el diagrama de secuencia de autenticación. Se puede observar como el administrador del servicio web, del portal Dashboard Application, es el que inicia la secuencia al querer acceder a las funcionalidades ofrecidas por parte del portal. El portal ofrece un formulario para que el administrador autentique sus credenciales, con el servicio. Cuando el administrador realiza el submit, comienza la secuencia.

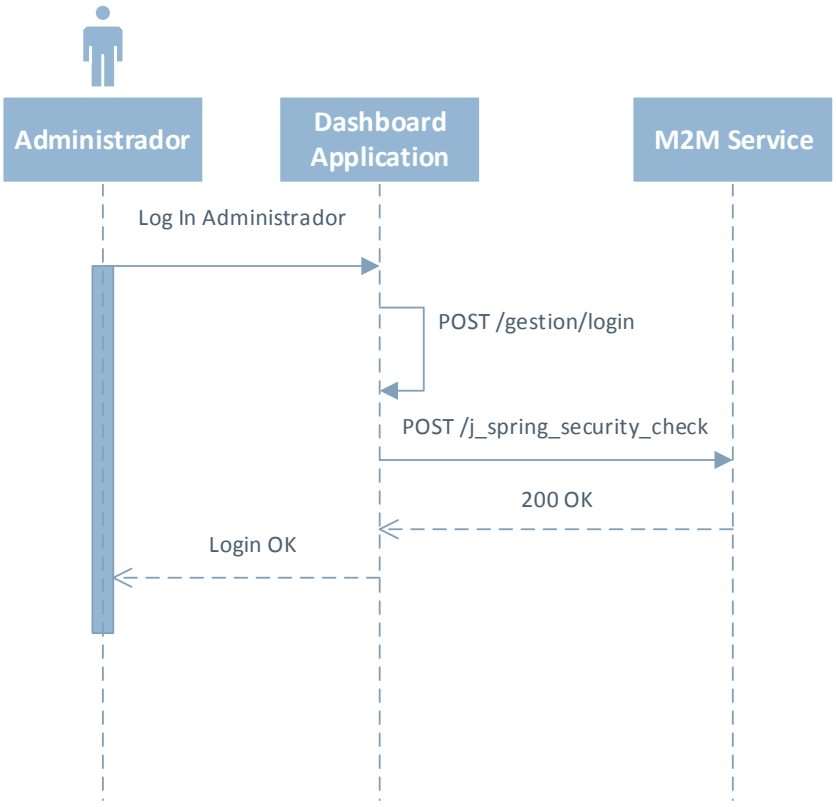

Figura 7. Diagrama de Secuencia Autenticación Administrador

El componente Dashboard Application, realiza una acción que tiene contenida en un controlador para realizar la llamada al servicio de autenticación que está integrado en el servicio M2MService.

M2MService recupera la información enviada desde el administrador y comprueba que las credenciales son válidas y genera los permisos suficientes para que el administrador pueda acceder a funcionalidades de gestión y administración ofrecidas por M2MService, y que de este modo, puedan realizarse peticiones HTTP de POST y DELETE, previamente bloqueadas a usuarios no autenticados que por tanto, no tienen permisos.

Como respuesta a la validez de las credenciales, M2MService manda una respuesta de un 200 OK según la regla HTTP para peticiones y respuestas, indicando que la autenticación y el registro del cliente utilizado por el administrador se han realizado correctamente. El portal Dashboard Application recibe esta respuesta y si ha sido válida, pasará a mostrar nuevas funcionalidades para el administrador, como son las pestañas de dispositivos y alertas, donde podrá gestionar los dispositivos y la configuración de las alertas que se recibirán a partir del registro de las medidas realizadas por los dispositivos.

El diagrama de secuencia para la acción de cerrar sesión de administrador se realiza de la misma forma que la autenticación pero varía el *endpoint* utilizado del servicio M2MService, siendo este /j_spring_security_logout.

AÑADIR NUEVO DISPOSITIVO

El diagrama de secuencia de la figura siguiente muestra el proceso del registro de un nuevo dispositivo al sistema, para que de este modo se puedan recoger los datos y registro de medidas realizado por un dispositivo.

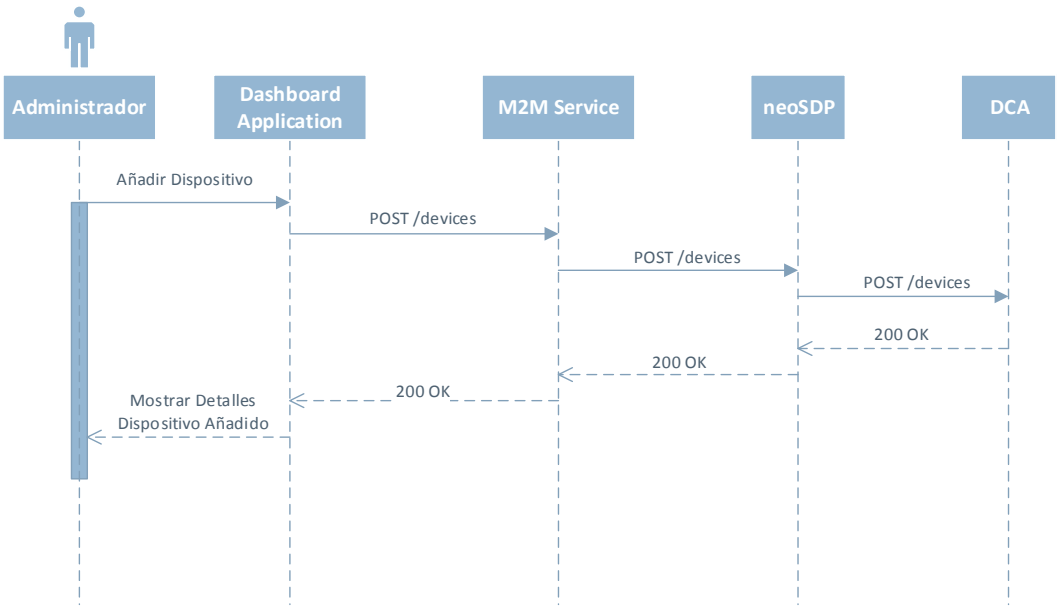

Figura 8. Diagrama de Secuencia Añadir Nuevo Dispositivo

El paso previo requerido a esta acción es la autenticación, ya que sin una autenticación previa del administrador, no va a poder acceder a la opción en el portal Dashboard Application.

El proceso de añadir un nuevo usuario comienza con el acceso por parte del administrador al formulario para el registro de un dispositivo, donde se indicará su localización, el modelo sobre el que se basa el dispositivo, una breve descripción y su identificador. Una vez rellenado el formulario necesario, el administrador debe hacer un *submit* para que se realice el registro del dispositivo en el DCA, de este modo se lo comunica al controlador que gestiona esta función en Dashboard Application.

Dashboard Application se encarga de realizar una petición POST, con los datos necesarios y rellenados previamente por el administrador, hacia el servicio creado M2MService.

M2MService se encarga de realizar esta petición POST añadiendo las credenciales válidas y comprobando contra neoSDP que el certificado que posee el proveedor de este servicio, que lo tiene contratado, es válido. Así se muestran las funcionalidades que ofrece neoSDP en cuanto a la exportación de servicios ofrecidos por Telefónica.

neoSDP se encarga de comunicarse y transferir la petición al servicio ofrecido por Telefónica, en este caso la API de M2M, para comunicarse con el DCA, y así quede registrado en sus servidores. neoSDP, como se ha comentado anteriormente en el análisis de la arquitectura, es el componente ofrecido por Telefónica para esta solución, mediante el balanceador de carga establecerá el camino más adecuado para lanzar la petición al otro componente de Telefónica que es partícipe en la solución, el DCA.

El DCA realizará el registro y, si se ha realizado correctamente, devolverá como respuesta un 200 OK. Esta respuesta debe de llegar hasta el componente Dashboard Application. Por tanto el recorrido que realizará la respuesta será el mismo que el que ha realizado la petición. neoSDP devolverá la respuesta al componente M2MService y este a su vez al portal. Como se indica en el diagrama de secuencia, el Dashboard Application mostrará los datos registrados por el DCA acerca del dispositivo. De esta manera el administrador puede visualizar en la aplicación web la lista de dispositivos que se encuentran registrados en el DCA asociados al Tercero que ha contratado el servicio. Además permitirá visualizar detalladamente los datos relacionados con cada dispositivo y un resumen de sus últimas medidas registradas.

El componente de Telefónica DCA es con el que el dispositivo registrado establecerá comunicación a través de las redes móviles para realizar el registro de las medidas realizadas por sí mismo y de este modo, además, a través del DCA se puede realizar un control y manejo del dispositivo.

Otras funcionalidades ofrecidas como es la función de borrar un dispositivo del registro de dispositivos, la metodología y secuencia realizada es similar a la secuencia aquí

descrita, de añadir un nuevo dispositivo. En el Anexo adjunto se pueden ver los diagramas de secuencia similares a este descrito como se comentaba, por ejemplo el diagrama de secuencia de borrar un dispositivo.

AÑADIR NUEVA ALERTA

En este diagrama de secuencia de la figura siguiente se muestra la funcionalidad de la creación de alertas para poder recibir en tiempo real si se alcanza un límite o un valor configurado previamente en la medida de algún tipo de elemento. Esto puede ser por ejemplo que se supere un valor preestablecido de temperatura, la configuración de que si se sobrepasa el valor de 30º C se reciba y se registre el mensaje de que se ha sobrepasado dicho nivel.

En este diagrama se muestra la funcionalidad de añadir una nueva alerta, posteriormente se mostrará cómo será recibida por parte de la aplicación web una alerta, y así el administrador o usuario podrá ver que alerta se ha producido.

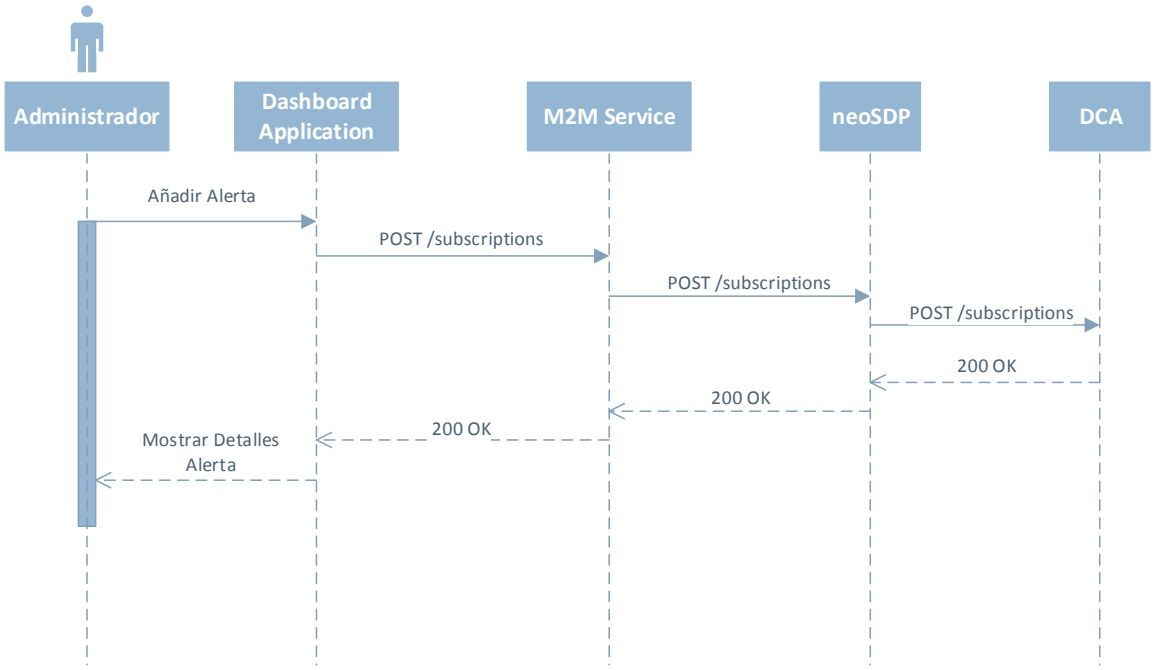

Figura 9. Diagrama de Secuencia Añadir Nueva Alerta

El proceso comienza cuando el administrador, como en el proceso anterior de registro de un nuevo dispositivo, se autentica. De igual manera, para la creación de nuevas alertas, solamente se permite la gestión y administración a un administrador validado contra el componente M2MService.

Una vez pasado el proceso de autenticación, como en el proceso anterior, se necesita rellenar el formulario de registro de la alerta. En este caso el procedimiento es distinto, es necesario indicar sobre qué tipo de medida se quiere realizar el control y sobre que dispositivos ya registrados se realizará el control. Una vez indicado el tipo de medida y los dispositivos, se indicará el valor de referencia y si la alerta se lanzará cuando sea mayor, menor o igual a ese valor de referencia. Una vez rellenado se procede a enviar la petición de registro al Dashboard Application, este se encargará de realizar una petición POST al componente M2MService especificando en el *endpoint* el tipo de acción que se quiere realizar, siendo ésta de tipo /subscription.

M2MService se encarga, de igual manera que en el diagrama de secuencia analizado anteriormente en el registro de un nuevo dispositivo, de adjuntar los credenciales necesarios y un certificado para validar que el servicio está asociado a un cliente confiable para Telefónica, que haya contratado el servicio anteriormente. Además, adjunta el *endpoint* para tener una entrada a la llegada de las alertas y así posteriormente poder indicar al Dashboard Application que se ha producido una alerta.

neoSDP se encarga de redirigir la petición hacia el *endpoint* correcto, el cual es el asignado para el DCA. El DCA realiza el registro de la nueva alerta, y si se ha realizado correctamente se devolverá un mensaje de respuesta con el código 200 OK.

Cuando la respuesta es recibida por M2MService, se lo comunica a Dashboard Application y esta actualiza la lista que muestra de las alertas suscritas.

Para realizar el borrado de una suscripción a una alerta, la secuencia que se sucede es exactamente igual que para este caso salvo el tipo de petición, que pasa a ser un DELETE, o en el caso de cambiar el estado de una suscripción a una alerta que el tipo de mensaje HTTP sería PUT para actualizar el estado. Diagramas de secuencia acerca estas operaciones se encuentran en el anexo adjunto a la memoria.

GENERAR INFORME GRÁFICO

En este diagrama de la figura 10 se describe una de las funcionalidades más importantes que ofrece el Dashboard Application y que es accesible por cualquier tipo de usuario, ya sea el administrador del servicio o cualquier cliente que desee informarse acerca de los datos recogidos por alguno de los dispositivos instalados en las estaciones de medida.

Mediante esta funcionalidad el usuario deberá seleccionar qué parámetro de medida quiere que se muestre en la gráfica, además deberá especificar si quiere mostrar el histórico de medidas completo, en una fecha determinada o en un rango de fecha determinado. Podrá seleccionar si quiere mostrar la media semanal, mensual o diaria.

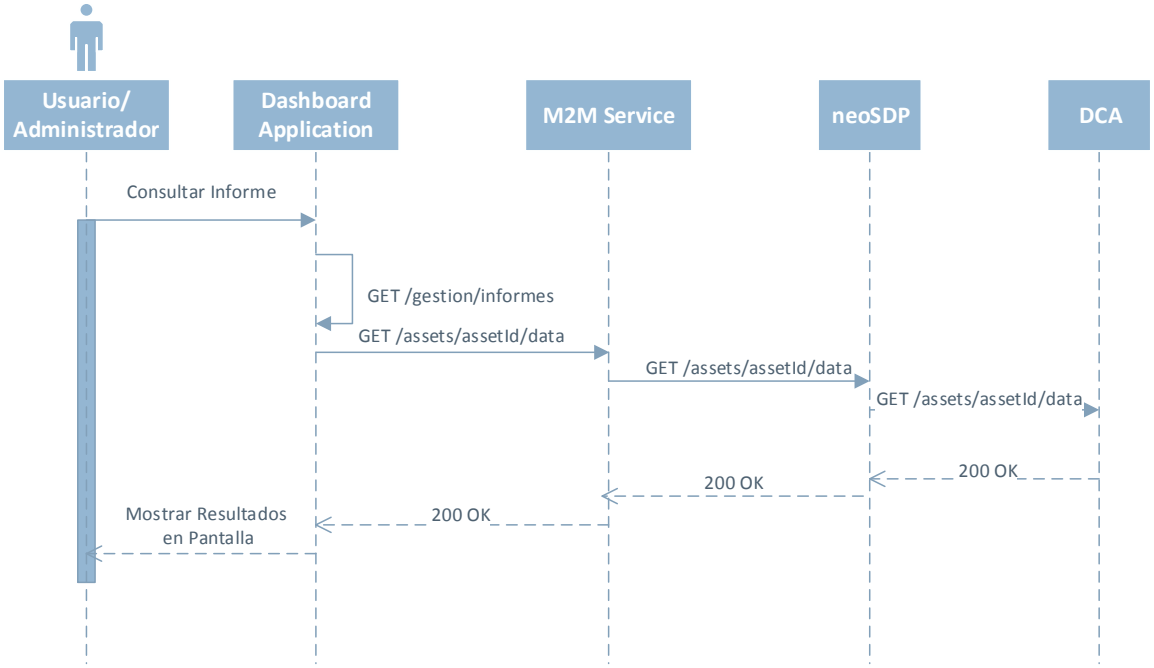

Figura 10. Diagrama de Secuencia Generar Informe Gráfico

Una vez seleccionados los parámetros necesarios para poder realizar la petición al servidor correctamente, el usuario lo comunica al Dashboard y este realiza una petición GET al M2MService de modo que le indica sobre que dispositivo se está pidiendo información, siendo su identificador la variable "assetId", que la rellena el propio Dashboard Application. Los parámetros de configuración de la consulta van añadidos como "queryparameters" en el *endpoint* asociado.

M2MService realiza la misma petición recibida hacía el DCA pasando a través de la plataforma neoSDP de la misma manera que lo que se ha comentado en los diagramas de secuencia anteriores.

Cuando el DCA recibe la petición realiza la búsqueda del registro del dispositivo indicado y filtra el histórico registrado en función de los parámetros de consulta, devuelve la petición y devuelve la información como texto en formato JSON y con un código de la respuesta, 200 OK.

El Dashboard Application recibe la respuesta y transforma los datos recibidos para que se puedan mostrar correctamente las medidas registradas en la gráfica de la web.

RECIBIR INFORMACIÓN DISPOSITIVOS REGISTRADOS

En este diagrama de secuencia de la siguiente figura se muestra como se ejecuta la secuencia de mostrar en la interfaz web del Dashboard Application la lista de dispositivos registrados en el DCA para el proveedor contratante de los servicios de Telefónica.

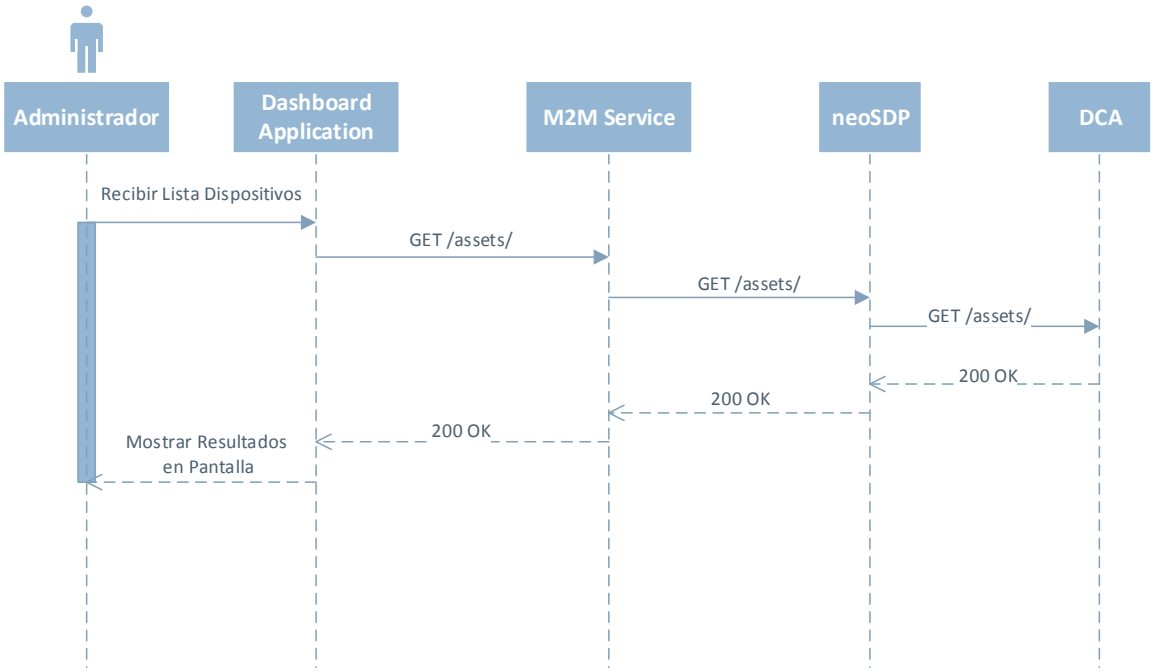

Figura 11. Diagrama de Secuencia para Recibir Lista de Dispositivos

Como se puede observar en el diagrama, la petición que se realiza es del tipo GET y se devuelve en formato JSON la lista con todos los dispositivos registrados y sus parámetros.

3.4 M2MSERVICE

El componente M2MService es un servicio REST creado para encapsular los servicios ofrecidos Telefónica de M2M a través de la plataforma neoSDP. Este componente ha sido desarrollado como parte de la solución propuesta.

Ha sido desarrollado en Java utilizando las librerías creadas por Spring para la creación de servicios WEB y REST además de las propias librerías de Java indicadas para estos servicios.

La funcionalidad principal que ofrece el componente es la utilización de los servicios por la API de M2M de Telefónica, además añade una capa de seguridad para el administrador externo de los servicios contratados de Telefónica y efectúa la autenticación necesaria para poder acceder a través de neoSDP a la API contratada por el externo. Por tanto, el componente M2MService está compuesto por dos elementos: la API de M2M y el servicio de autenticación implementado mediante las herramientas ofrecidas por Spring Security.

3.4.1 DEFINICIÓN DE ELEMENTOS

Se va a llevar a cabo la definición de los elementos utilizados en el desarrollo e implementación de los servicios. De esta manera serán familiares y reconocibles para cuando sean referenciados posteriormente.

MODEL (MODELO)

Este elemento es definido por el DCA y es utilizado para crear un perfil de los dispositivos que se vayan a utilizar. La definición de un modelo se realiza a partir de las herramientas ofrecidas por el DCA y es accesible a través del servicio REST ofrecido por M2MService. Es un elemento imprescindible para la creación de un Asset/Device.

ASSET (ELEMENTO)

Este elemento es la definición de un elemento físico. En el caso de la solución los devices/assets son simulados y no están registrando un dispositivo real. A través del servicio REST se puede acceder a la información de los assets, de igual manera se pueden crear o borrar sus referencias creadas.

DEVICE (DISPOSITIVO)

Como entidad tiene el mismo valor y hace referencia al mismo elemento que un Asset, sin embargo a la hora de consultar y recibir los datos desde el servicio rest, los datos mostrados se muestran diferentes y con distintos parámetros. Un Device referencia al mismo elemento que un Asset, es decir, que toda consulta acerca de datos registrados corresponderá al mismo sea consultado a través de devices o de assets además de que si se borra un device también se está borrando un asset.

SENSOR

Es el elemento que cada componente de un device utilizado para realizar las medidas. En el modelo se especifica qué tipo de sensores son utilizados por los dispositivos que sigan un modelo.

ALERT (ALERTA)

Este elemento es el utilizado para que el usuario conozca si se ha alcanzado un valor de riesgo en la medida de un parámetro concreto. A través del servicio REST se permite la creación de reglas para generar alertas cuando se alcancen o se sobrepasen valores límite, permitiendo la aplicación de estas reglas a los dispositivos que quiera el usuario. El usuario se entiende como el administrador del servicio contratado.

ESTACIÓN

Con el nombre de estación se refiere al conjunto de dispositivos localizados en una misma posición. Es decir, a todos los dispositivos que estén registrados en una misma posición (Latitud y longitud) pertenecerán a la misma estación. Esta designación se realiza para que el demostrador sea una simulación lo más cercana a la realidad, y pueda identificarse con la instalación de estaciones base en zonas verdes a lo largo de las ciudades.

A continuación se describirá cada elemento desglosando las partes que componen cada uno de ellos.

3.4.2 SERVICIO DE AUTENTICACIÓN

El servicio de autenticación se ha implementado para ofrecer un bloqueo a usuarios independientes al contratante de los servicios ofrecidos por Telefónica, de modo que no puedan acceder a servicios de gestión y mantenimiento ofrecidos. Creándose un usuario "Administrador" se ofrece exclusivamente los servicios de gestión y mantenimiento. Mediante una autenticación de los credenciales del usuario, se podrá acceder a estos servicios bloqueados previamente y quedando ocultos en la interfaz web de Dashboard Application a los usuarios que no están autorizados.

Este servicio desarrollado es accesible tanto mediante, como se ha mencionado anteriormente, el Dashboard Application y desde cualquier cliente REST que se desee utilizar siendo los *endpoint*s accesibles los siguientes:

-Inicio de Sesión:

```
{baseURI}/M2MService/j_spring_security_check
```

-Cierre de sesión

`{baseURI}/M2MService/j_spring_security_logout`

A partir del tutorial [27] sobre utilización e implementación de servicios desarrollados mediante las librerías de Spring se ha tomado como punto de partida para la implementación en el componente M2MService el servicio de autenticación.

Para el desarrollo de este servicio se ha añadido un conjunto de librerías externas a las librerías por defecto de Java, spring-security-core, spring-security-web y spring-security-config.

La estructura de clases creadas para esta implementación es la siguiente:

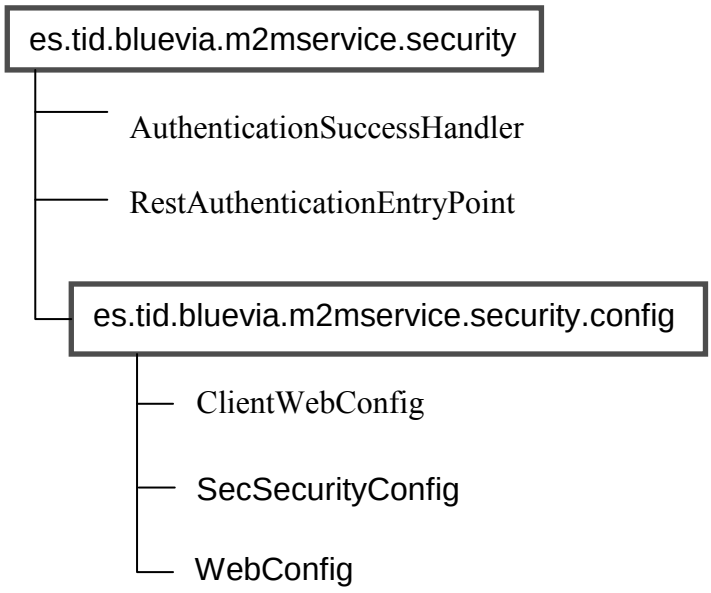

Figura 12. Estructura de Clases Servicio de Autenticación

Mediante estos paquetes de clases se ha realizado la implementación. En el paquete interno, "config" se han desarrollado tres clases cada una de ellas específica para la configuración de los elementos necesarios de Spring, de este modo, al necesitarse un servicio genérico de autenticación, se ha optado por la configuración por defecto de Spring, sin aplicar ninguna modificación en su configuración. Solamente se indica sobre quequé elementos de la estructura del servicio REST se quiere realizar el escaneo para aplicar el servicio de seguridad para limitar el acceso a funcionalidades si no se ha realizado la autenticación oportuna. En este caso se indica sobre que componente se quiere realizar el control siendo `es.tid.bluevia.m2mservice.rest.` También en la clase SecSecurityConfig se indica el fichero "spring-security.xml" en el que se indican todos los parámetros y configuraciones de seguridad que se desean aplicar, siendo este fichero el siguiente:

```
<?xml version="1.0" encoding="UTF-8"?>
<beans:beans xmlns="http://www.springframework.org/schema/security"
	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:beans="http://www.springframework.org/schema/beans"
	xmlns:sec="http://www.springframework.org/schema/security"
	xsi:schemaLocation="
	http://www.springframework.org/schema/security
	http://www.springframework.org/schema/security/spring-security-3.2.xsd
	http://www.springframework.org/schema/beans
	http://www.springframework.org/schema/beans/spring-beans-4.0.xsd">

	<http entry-point-ref="restAuthenticationEntryPoint"
		use-expressions="true">
		<intercept-url pattern="/M2MRestService/alerts" access="permitAll"
			method="POST" />
		<intercept-url pattern="/M2MRestService/**"
access="isAuthenticated()"
			method="POST" />
		<intercept-url pattern="/M2MRestService/**"
access="isAuthenticated()"
			method="DELETE" />


		<sec:form-login authentication-success-handler-ref="SuccessHandler"
			authentication-failure-handler-ref="FailureHandler" />

		<logout />
	</http>

	<beans:bean id="SuccessHandler"

	class="es.tid.bluevia.m2mservice.security.AuthenticationSuccessHandler"
/>
	<beans:bean id="FailureHandler"

	class="org.springframework.security.web.authentication.SimpleUrlAuthentic
ationFailureHandler" />

	<authentication-manager alias="authenticationManager">
		<authentication-provider>
			<user-service>
				<user name="admin" password="1234"
authorities="ROLE_ADMIN" />
				<user name="user" password="1234"
authorities="ROLE_USER" />
			</user-service>
		</authentication-provider>
	</authentication-manager>

</beans:beans>
```

Mediante este fichero se realiza la configuración del servicio de seguridad. De arriba a abajo se puede realizar una pequeña vista de qué se ha configurado rápidamente. Primero, se ve en los atributos de <beans> qué versión de las librerías de Spring se ha

utilizado para los esquemas. Segundo, en la referencia restAuthenticationEntryPoint, se indica qué *path* se quiere interceptar para el servicio, siendo este el servicio REST ofrecido por el componente M2MService. Se quiere bloquear toda entrada mediante métodos POST y los métodos DELETE al servicio REST, salvo al *endpoint* /alerts porque es un servicio que será accedido desde el DCA si se cumple los requisitos para mandar una alerta al cliente.

Por tanto, el acceso a recursos del servicio REST que impliquen alguna modificación en los datos será limitado a usuarios previamente autenticados contra el servicio.

Los siguientes parámetros de configuración en el XML se refieren al controlador de autenticación previamente configurado con las clases java y cómo actuará en caso de una autenticación correcta o incorrecta.

Por último, queda el control de usuarios, que actualmente se realiza en este fichero, pero que como trabajo futuro se puede realizar el control de usuarios mediante el uso de una base de datos donde registrar información de los administradores registrados además de realizar la consulta de sus datos para validar los datos recibidos y confirmar la autenticación.

INICIO DE SESIÓN

Según el componente desarrollado por Spring, el servicio de autenticación es accesible mediante un método POST al *endpoint* referido anteriormente. Como parámetros se añaden dos: j_username y j_password. De este modo el *endpoint* completo queda de la siguiente manera:

```
{baseURI}/M2MService/j_spring_security_check?j_username=
USUARIO&j_password=CONTRASEÑA
```

Como se indicó anteriormente en los diagramas de secuencia en el proceso de inicio de sesión, una vez que se ha realizado la petición a M2MService, éste contestará con un 200 OK si se ha realizado correctamente la autenticación y un 401 Unauthorized si no ha sido válida.

El servicio desarrollado con Spring Security registra una cookie con un identificador de la máquina y navegador o servicio utilizado para realizar la comunicación con el servicio M2MService de modo que se queda registrado y así ya puede acceder a recursos ofrecidos por el componente para realizar la creación, modificación o borrado de datos.

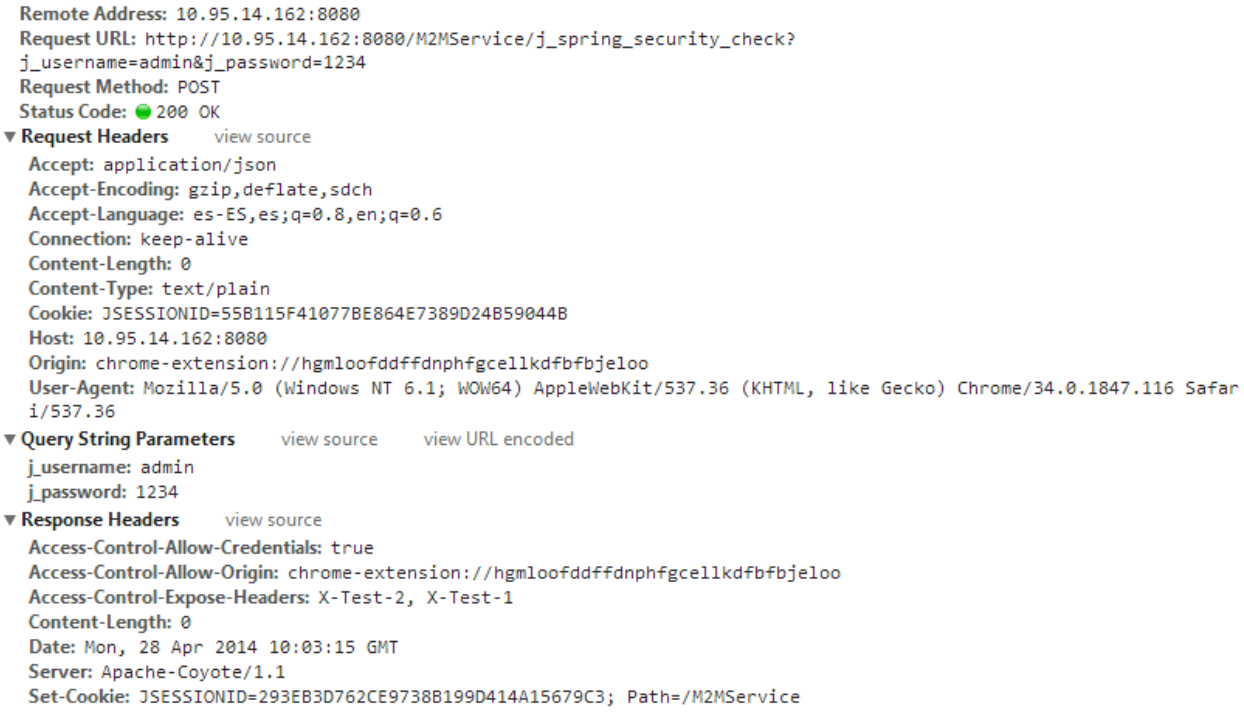

Figura 13. Captura Cabeceras Servicio de Autenticación

En la figura 13 se muestra una captura donde se pueden apreciar los parámetros enviados y la respuesta recibida, de modo que se realiza el salvado de la cookie que es necesaria para que el servicio M2MService controle la validez de las peticiones recibidas. Se observa que la petición es mediante un método POST.

CIERRE DE SESIÓN

El cierre de sesión se realiza del mismo modo que en el apartado anterior, aunque en este caso sin enviar los credenciales de usuario. Simplemente se debe adjuntar en la cabecera de la petición POST la cookie registrada anteriormente, de este modo el servicio de autenticación de M2MService borrará los permisos para este usuario y así si se desea volver a acceder a algún recurso de modificación o borrado de M2MService sea necesario volver a autenticarse.

El *endpoint* es el indicado anteriormente:

`{baseURI}/M2MService/j_spring_security_logout`

Y un ejemplo de este Servicio es la siguiente captura, que como se observa, si se realiza correctamente el servicio devuelve un 302 Found.

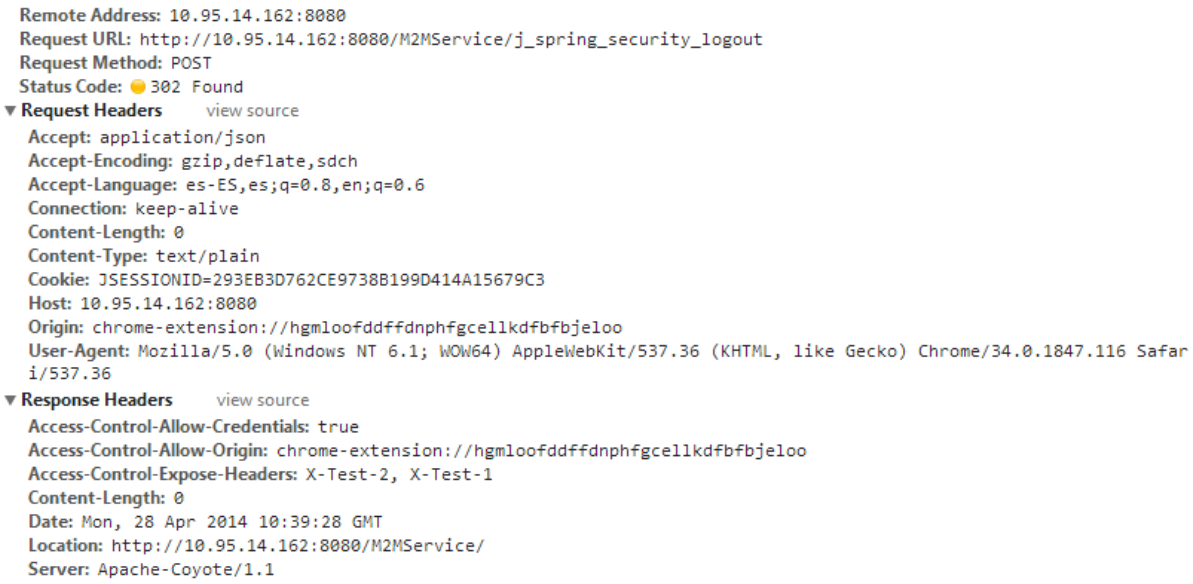

Figura 14. Cierre de Sesión

3.4.3 API REST

El servicio REST implementado hace uso de las librerías Java disponibles por defecto y las desarrolladas por Spring para la creación de servicios Web. En este caso estas librerías son las siguientes: JAX-WS, y Spring Web. El servicio ha sido desarrollado en Java y ha sido desarrollado como servicio web para que posteriormente sea desplegado en un servidor de aplicaciones, en este caso se ha utilizado un Apache Tomcat 6. La estructura del servicio sigue la misma estructura básica de los servicios REST, un conjunto de recursos disponibles y accesibles mediante peticiones GET, POST o DELETE a partir de la definición de unos *endpoint*s además de la definición de los parámetros y atributos de consultas en algunos casos.

Para la descripción del servicio desarrollado, inicialmente se comenzará desde una visión más externa hasta una visión más interna de las funcionalidades que ofrece cada recurso desarrollado.

DEFINICIÓN

Para la definición de un servicio REST se suelen utilizar servicios para definir de una forma simple e intuitiva los recursos ofrecidos en el servicio, de manera que se pueda identificar todo lo que ofrece el servicio REST. Para ello, uno de estos servicios disponibles es WADL, un recurso disponible implementado en un fichero XML para describir los servicios de un servicio Web, principalmente servicios REST, como es este caso. WADL es el servicio equivalente de los servicios REST a WSDL de los servicios implementados en SOAP.

En este caso, la definición WADL del servicio REST de M2MService es la siguiente:

```
<application xmlns="http://wadl.dev.java.net/2009/02" xmlns:xs="http://www.w3.org/2001/XMLSchema">
	<grammars/>
	<resources base="http://10.95.14.162:8080/M2MService/M2MRestService">
		<resource path="/">
			<resource path="alerts">
				<method name="GET">
					<response>
						<representation mediaType="application/json"/>
					</response>
				</method>
				<method name="POST">
					<request>
						<representation mediaType="text/xml">
							<param name="request" style="plain"
type="xs:string"/>
						</representation>
					</request>
					<response status="204"/>
				</method>
			</resource>
			<resource path="assets">
				<method name="GET">
					<response>
						<representation mediaType="application/json"/>
					</response>
				</method>
				<method name="POST">
					<request>
						<representation mediaType="application/json"/>
					</request>
					<response>
						<representation mediaType="application/json"/>
					</response>
				</method>
			</resource>
			<resource path="assets/{assetId}">
				<param name="assetId" style="template" type="xs:string"/>
				<method name="DELETE">
					<request/>
					<response>
						<representation mediaType="application/json"/>
					</response>
				</method>
				<method name="GET">
					<request/>
					<response>
						<representation mediaType="application/json"/>
					</response>
				</method>
			</resource>
			<resource path="assets/{assetId}/data">
				<param name="assetId" style="template" type="xs:string"/>
				<method name="GET">
					<request>
						<param name="interval" style="query" type="xs:string"/>
						<param name="sortBy" style="query" type="xs:string"/>
						<param name="attribute" style="query" type="xs:string"/>
						<param name="value" style="query" type="xs:string"/>
						<param name="scope" style="query" type="xs:string"/>
						<param name="property" style="query" type="xs:string"/>
						<param name="param" style="query" type="xs:string"/>
						<param name="name" style="query" type="xs:string"/>
						<param name="limit" style="query" type="xs:string"/>
					</request>
					<response>
						<representation mediaType="application/json"/>
					</response>
				</method>
			</resource>
			<resource path="devices">
				<method name="GET">
					<response>
						<representation mediaType="application/json"/>
					</response>
				</method>
				<method name="POST">
					<request>
						<representation mediaType="application/json">
							<param name="request" style="plain"
```

```
type="xs:string"/>
                                        </representation>
                                </request>
                                <response>
                                        <representation mediaType="application/json"/>
                                </response>
                        </method>
                </resource>
                <resource path="devices/{deviceId}">
                        <param name="deviceId" style="template" type="xs:string"/>
                        <method name="GET">
                                <request/>
                                <response>
                                        <representation mediaType="application/json"/>
                                </response>
                        </method>
                </resource>
                <resource path="devices/{deviceId}/data">
                        <param name="deviceId" style="template" type="xs:string"/>
                        <method name="GET">
                                <request>
                                        <param name="interval" style="query" type="xs:string"/>
                                        <param name="sortBy" style="query" type="xs:string"/>
                                        <param name="attribute" style="query" type="xs:string"/>
                                        <param name="value" style="query" type="xs:string"/>
                                        <param name="scope" style="query" type="xs:string"/>
                                        <param name="property" style="query" type="xs:string"/>
                                        <param name="param" style="query" type="xs:string"/>
                                        <param name="name" style="query" type="xs:string"/>
                                        <param name="limit" style="query" type="xs:string"/>
                                </request>
                                <response>
                                        <representation mediaType="application/json"/>
                                </response>
                        </method>
                </resource>
                <resource path="models">
                        <method name="GET">
                                <response>
                                        <representation mediaType="application/json"/>
                                </response>
                        </method>
                        <method name="POST">
                                <request>
                                        <representation mediaType="application/json"/>
                                </request>
                                <response>
                                        <representation mediaType="application/json"/>
                                </response>
                        </method>
                </resource>
                <resource path="models/{modelId}">
                        <param name="modelId" style="template" type="xs:string"/>
                        <method name="GET">
                                <request/>
                                <response>
                                        <representation mediaType="application/json"/>
                                </response>
                        </method>
                </resource>
                <resource path="models/{modelId}/data">
                        <param name="modelId" style="template" type="xs:string"/>
                        <method name="GET">
                                <request>
                                        <param name="interval" style="query" type="xs:string"/>
                                        <param name="sortBy" style="query" type="xs:string"/>
                                        <param name="attribute" style="query" type="xs:string"/>
                                        <param name="value" style="query" type="xs:string"/>
                                        <param name="scope" style="query" type="xs:string"/>
                                        <param name="property" style="query" type="xs:string"/>
                                        <param name="param" style="query" type="xs:string"/>
                                        <param name="name" style="query" type="xs:string"/>
                                        <param name="limit" style="query" type="xs:string"/>
                                </request>
                                <response>
                                        <representation mediaType="application/json"/>
                                </response>
                        </method>
                </resource>
                <resource path="status">
                        <method name="GET">
                                <response>
                                        <representation mediaType="application/json"/>
```

```
                                        </response>
                                </method>
                                <method name="POST">
                                        <request>
                                                <representation mediaType="application/json">
                                                        <param name="request" style="plain"
type="xs:string"/>
                                                </representation>
                                        </request>
                                        <response>
                                                <representation mediaType="application/json"/>
                                        </response>
                                </method>
                        </resource>
                        <resource path="subscriptions">
                                <method name="GET">
                                        <response>
                                                <representation mediaType="application/json"/>
                                        </response>
                                </method>
                                <method name="POST">
                                        <request>
                                                <representation mediaType="application/json">
                                                        <param name="request" style="plain"
type="xs:string"/>
                                                </representation>
                                        </request>
                                        <response>
                                                <representation mediaType="application/json"/>
                                        </response>
                                </method>
                        </resource>
                        <resource path="subscriptions/{subscriptionId}">
                                <param name="subscriptionId" style="template" type="xs:string"/>
                                <method name="GET">
                                        <request/>
                                        <response>
                                                <representation mediaType="application/json"/>
                                        </response>
                                </method>
                        </resource>
                        <resource path="subscriptions/{subscriptionsId}">
                                <param name="subscriptionsId" style="template" type="xs:string"/>
                                <method name="DELETE">
                                        <request/>
                                        <response>
                                                <representation mediaType="application/json"/>
                                        </response>
                                </method>
                        </resource>
                </resource>
        </resources>
</application>
```

Inicialmente, si se analiza el WADL mostrado se pueden observar distintos parámetros.

Primero, cada recurso que se ofrece en el servicio REST se muestra con el elemento "*resource*", indicándose cuál es el *endpoint* mediante el que se accede al servicio con el atributo "*path*".

Dentro del "*path*" el contenido marcado por {} es referido a un identificador que se utilizará para realizar la consulta. Además, cuando se accede a servicios que son de consulta de datos es posible que esté disponible la inclusión de algunos parámetros de consulta para filtrar el contenido a consultar, estos parámetros vienen referidos en el elemento "*param*" dentro del elemento "*request*".

En el WADL se especifica toda la información necesaria para conocer qué tipo de contenido consume el servicio REST, siendo en este caso en su mayoría contenido en JSON. Por último, otro elemento importante a reseñar es el tipo de método utilizado en la

petición HTTP, siendo posible en la mayoría de recursos, métodos GET, POST y DELETE.

Este servicio REST tiene como finalidad encapsular la API ofrecida por Telefónica y ofrecerla además de incluir de manera transparente el tratamiento de la capa de seguridad de neoSDP. Este servicio ha sido creado para facilitar la implementación de nuevos servicios al proveedor externo que contrate el servicio a Telefónica. De este modo, con la instalación de un certificado que autentique contra neoSDP al contratante y el despliegue del servicio web creado será suficiente para que a partir de este componente se puedan crear aplicaciones M2M que hagan uso del componente, como ejemplo es la aplicación web creada en esta solución Dashboard Application. A continuación se hará una descripción de los recursos ofrecidos indicando la funcionalidad que tiene y su comportamiento en la figura 15.

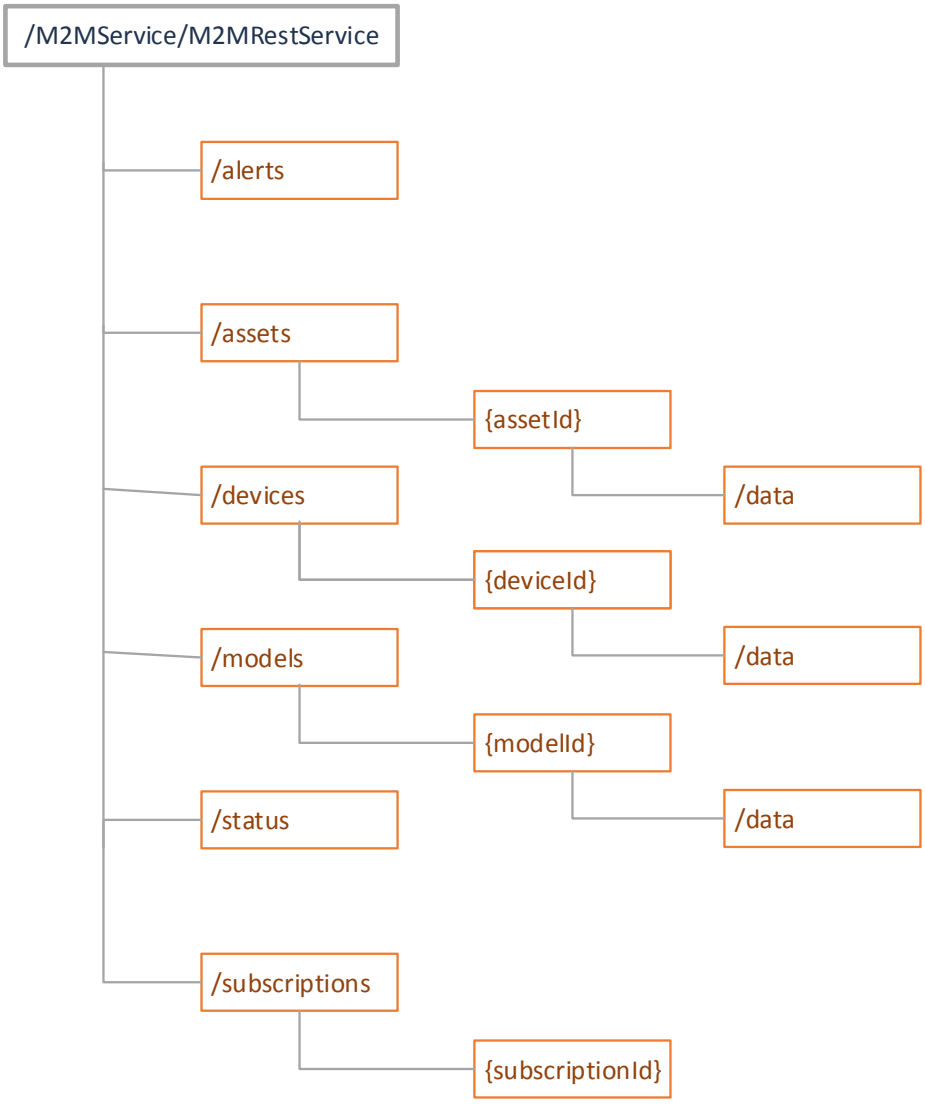

Figura 15. Servicio REST

El acceso a los recursos creados en el servicio REST debe de ser de la siguiente manera:

- Métodos HTTP: GET, POST o DELETE
- Cuerpo del mensaje de petición: JSON (si es necesario)

{HOST}/M2MSERVICE/M2MRESTSERVICE/ALERTS

Este recurso se ofrece en dos métodos, GET y POST. En el primer caso, en el caso del acceso al recurso mediante un GET se realiza para recibir las alertas que se han producido a partir de las reglas establecidas previamente. El servicio M2MService tiene un elemento que es una instancia global, de modo que en el servicio se almacenan las alertas recibidas para poder una vez que se acceda a ellas mediante un GET poder mostrarlas en una interfaz visible al usuario, por ejemplo mediante el componente Dashboard Application creado en la solución.

En el caso del método POST se ha implementado para dar soporte al DCA, ya que según las reglas establecidas en el DCA, que en el caso de crear una alerta se debe proporcionar un *endpoint* disponible para enviar información en tiempo real si se produce una alerta. En este caso, la información enviada por el DCA sigue el formato del Lenguaje SensorML que provee en formato XML una semántica específica en el intercambio de información a través de sensores y actuadores. Este lenguaje ha sido desarrollado por el consorcio OGC (Open Geospatial Consortium). Posteriormente en el apartado de resultados se mostrará un ejemplo de los datos recibidos por el servicio desde el DCA.

{HOST}/M2MSERVICE/M2MRESTSERVICE/ASSETS

Este recurso ofrece tres tipos de métodos de comunicación, GET, POST y DELETE. Primero, cabe resaltar que solo es accesible sin una autenticación previa contra el servicio de autenticación de M2MService el método GET. Sobre el uso del método GET, existen varias opciones de comunicación tanto en cuanto *endpoint*s se ofrecen. En el caso más externo es el acceso a "/assets", que como resultado se devuelve en formato JSON una lista de los *assets* definidos en el DCA y sus últimos registros de cada parámetro. Otra forma de acceder en este caso es a la información de un *asset* concreto, siendo identificado con el "assetId". Su *endpoint* es "/assets/{assetId}" y se accede a la misma información que se ha recibido en el caso anterior, pero ahora solo siendo del asset indicado. Como último tipo de acceso GET es a través del *endpoint* "/assets/{assetId}/data" a partir del cual se podrá recibir todos los valores de las medidas registradas por un *asset*, además con la posibilidad de añadir parámetros para realizar la "query" y así filtrar la información que se quiere consultar. Los atributos disponibles para realizar la consulta son los siguientes:

- interval
- attribute
- sortBy
- limit
- value
- scope
- property
- param
- name

El registro de un *asset* se hace mediante el método POST al *endpoint* "/assets", y dentro del cuerpo del mensaje se manda la información a registrar de un *asset* en formato JSON. Siguiendo la definición del documento de la API de M2M de Telefónica [28] se puede reseñar la forma que tendría el cuerpo de la petición:

```
{
        "name": "DeviceIdentifierAsPublished(System in SensorML)",
        "concentrator": "ConcentratorNameIfAssociated",
        "isConcentrator": "true/false",
        "notificationStatus": "NotifyOrNotNotify",
        "asset": {
                "name": "AssetNameProvisionedByUser", "description":
                "AssetDescriptionProvisionedByUser",
                "type": "AssetTypeDefinedByUser"
                },
        "group": "GroupByName",
        "UserProps": [
                {
                "name": "profile",
                "value": "ValueProvisionedByUser"
                },
                {
                "name": "profile",
                "value": "ValueProvisionedByUser"
                }
                ],
        "DeviceProps": {
                "manufacturer": "ValueByUserOrDevice",
                "model": "ValueByUserOrDevice",
                "version": "ValueByUserOrDevice",
                "serialNumber": "ValueByUserOrDevice"
                },
        "location": {
                "latitude": "LatitudeProvidedByUser",
                "longitude": "LongitudeProvidedByUser",
                "altitude": "AltitudeProvidedByUser"
                }
}
```

Este ejemplo es mostrado en la documentación de Telefónica. En el apartado de resultados, posteriormente, se mostrará un ejemplo de uso del registro de un asset.

Por último, la última operación disponible es DELETE, donde a partir de un assetId se indica la referencia del asset que se quiere borrar de los datos almacenados, de modo que cuando se ejecuta esta operación se borrarán todos los datos asociados al identificador provisto.

{HOST}/M2MSERVICE/M2MRESTSERVICE/DEVICES

Este recurso ofrece las mismas funcionalidades que el del apartado anterior, "/assets" ya que como se ha comentado anteriormente, device y asset están referidos a un mismo elemento según las especificaciones en la API que ofrece el DCA. De modo que ofrece operaciones GET, POST y DELETE. Están implementados los mismos ajustes de seguridad que en el caso anterior siendo solo accesibles POST y DELETE si se ha llevado una autenticación válida previa. Las operaciones que se llevan a cabo mediante peticiones GET devuelven información similar que con los assets, aunque ahora si se quiere acceder a la información de un device/asset concreto, se debe de incluir su identificador, no su nombre. También se puede realizar la consulta de los datos medidos.

La operación POST se utiliza para crear nuevos devices y sigue la misma forma que el elemento JSON referenciado en el apartado "/assets".

Mediante la operación DELETE se borrarán los datos asociados a un deviceId del DCA.

{HOST}/M2MSERVICE/M2MRESTSERVICE/MODELS

Mediante el *endpoint* "/models" se ofrece operaciones GET, POST y DELETE. Este recurso ofrece la posibilidad de crear modelos que posteriormente son utilizados para crear Assets/Devices.

En la creación de un Model, se asigna los sensores que van a llevar los dispositivos que sigan este modelo, de modo que se le asignan qué valores se van a medir y que otro tipo de parámetros se van a monitorizar, como puede ser nivel de batería o nivel de cobertura de red del dispositivo. Esta operación se realiza mediante un método POST y con la previa autenticación de un usuario válido con el componente. En el componente Dashboard Application no se externaliza esta operación ya que es realizada desde los componentes que ofrece Telefónica, y se deja esta función como trabajo futuro. Un ejemplo de la creación de un Model, según la documentación de la API M2M de Telefónica es la siguiente:

```
{
        "name": "ModelSP3",
        "capabilities": [{
                "name": "Temperatura",
                "property": "temperature",
                "format": {
                        "name": "Temperatura",
                        "alias": "t",
                        "phenomenon": "BasicPhenomenonFromIDASPhenomena",
                        "type": "Quantity",
                        "uom": "celsius"
```

```
		},

	},
	{
		“name”: "Humedad",
		“property”: "relativeHumidity",
		“format”: {
			“name”: "Humedad",
			“alias”: "h",
			“phenomenon”: “BasicPhenomenonFromIDASPhenomena”,
			“type”: "Quantity",
			“uom”: "percent"
		}
	}],
	“parameters”: [{
		“name”: "BatteryLevel",
		“property”: "battery",
		“format”: {
			“alias”: "bl",
			“phenomenon”: “BasicPhenomenonFromIDASPhenomena”,
			“type”: "Quantity",
			“uom”: "percent"
		},

	},
	{
		“name”: "Alarm",
		“property”: "event",
		“format”: {
			“alias”: "e",
			“phenomenon”: “BasicPhenomenonFromIDASPhenomena”,
			“type”: "Boolean"
		}
	}],
	“commands”: [{
		“name”: "GetTime",
		“property”: "MyNameSpace:1.0:GetTime",
		“parameter”: {
			“type”: “Quantity”,
“phenomenon”: “BasicPhenomenonFromIDASPhenomena”,
“uom”: "seconds"
},
},
{
“name” : "Reset",
“property” : "MyNameSpace: 1.0: Reset",
}],
"UserProps": [
{
```

"name": "profile",
"value": "ValueProvisionedByUser"
},
{
"name": "owner"
}
],
“type” : “ThisFieldWillBeTranslateTo asset.type”
}

La operación GET devuelve como contenido el conjunto de todos los *Models* creados. Si se accede a través de un identificador de un modelo, “/models/{modelId}” se accede al contenido de un modelo concreto. La información devuelta es el conjunto de sensores que pueden ser utilizados además de los parámetros registrados en la creación previa del modelo.

Por último, la operación DELETE se borrará la información almacenada asignada a un identificador de un modelo.

{HOST}/M2MSERVICE/M2MRESTSERVICE/STATUS

Esta operación está creada para comprobar si el servicio está activo y desplegado correctamente en el servidor de aplicaciones. Mediante la operación GET se devuelve un 200 OK si está estable, y si no está activo y funcionando correctamente, devolverá un 404 (No Encontrado) ya que al no estar activo no será accesible la API REST creada.

{HOST}/M2MSERVICE/M2MRESTSERVICE/SUBSCRIPTIONS

El recurso “/subscriptions” ofrece operaciones GET, POST y DELETE. Este recurso es el utilizado para crear alertas y alarmas a partir de las medidas realizadas. Permite la asignación de reglas de control de las medidas realizadas, por ejemplo asignando un valor >, < o = a la medida de un parámetro además de asociarlo a los dispositivos que se desee. El usuario recibirá notificaciones a partir de los mensajes recibidos en una url que debe de ofrecer como “*callback*” donde recibirá en formato SensorML las notificaciones.

Como funcionalidad añadida a este servicio, está disponible la recepción de SMS por parte del Administrador, que será el que controlará todo el registro y mantenimiento de las alertas y suscripciones realizadas. Los clientes del proveedor podrán acceder a estos valores si éste las oferta de manera pública. En el caso de este proyecto realizado se ofrece de manera pública las alertas recibidas, de manera que cualquiera que consulte el portal ofrecido en Dashboard Application podrá conocer en tiempo real si se produce algún valor anómalo en función de las reglas establecidas.

La creación de estas suscripciones y reglas se realiza mediante la operación POST con su respectiva autenticación como en todos los recursos anteriores. El ejemplo ofrecido por la documentación de Telefónica es el siguiente:

```
{
   "name": "SubscriptionNameProvidedByUser",
   "description": "GroupDescriptionProvidedByUser",
   "type": "RegisterOrObservation",
   "resource": {
      "groups": [
         {
            "name": "groupName"
         },
         {
            "name": "groupName"
         }
      ],
      "assets": [
         {
            "name": "assetName"
         },
         {
            "name": "assetName"
         }
      ]
   },
   "filter": {
      "property": "temperature",
      "operator": "=",
      "max": 25,
      "min": "MinValue"
   },
   "notifyURI": "http: //host: 34/App",
   "validTime": "WhenSubscriptionIsNotValid"
}
```

Para acceder a los valores de las reglas creadas se realiza mediante un GET y finalmente si se quiere realizar el borrado de alguna de las reglas registradas se utiliza la operación DELETE mediante el uso del identificador de la suscripción hacia el *path* "/subscriptions/{subscriptionId}".

3.5 DASHBOARD APPLICATION

El componente llamado Dashboard Application es el componente que se sitúa como el servicio más cercano al usuario de la solución propuesta, es el que ofrece la interfaz gráfica al usuario y al administrador y se ha desarrollado con la intención de ser un

demostrador de las capacidades ofrecidas y desarrolladas por Telefónica, como se ha comentado en apartados anteriores.

Figura 16. Interfaz Dashboard Application

Es el componente que el usuario final tendrá que manejar, ya sea como administrador o como cliente anónimo del proveedor del servicio el que ofrece al público los servicios de Telefónica de M2M que han sido contratados y son dirigidos a su explotación.

La creación de este componente se ha desarrollado siguiendo el patrón de arquitectura software Modelo Vista Controlador (MVC) adaptado a la Web. Este patrón consiste en la separación de las funcionalidades ofrecidas por la aplicación de la interfaz de usuario. Se realiza una separación en tres elementos, el Modelo, la Vista y el Controlador. El Modelo es la representación de la información del sistema, controla el acceso a la información y realiza todo lo relacionado con su gestión. La Vista es el elemento encargado de visualizar de una manera que el usuario pueda entender la información recibida desde el Modelo, es la llamada más comúnmente como Interfaz de Usuario. Por último, el Controlador es el elemento que realiza la parte más compleja, controla los eventos que se producen en la Vista y es el encargado de comunicárselo al Modelo, es decir, su función, es de intermediario entre el Modelo y la Vista, manipula la información para que sea entendible por cada uno de los elementos.

El componente se ha desarrollado separando cada funcionalidad ofrecida en la interfaz web como un controlador y una vista, es decir, el componente queda formado por

un conjunto de controladores establecidos para cada una de las funcionalidades y vistas disponibles. Cada pestaña que se muestra en la barra de menú corresponde a un controlador creado junto con su vista.

Como modelos se han creado los objetos java llamados "*beans*", utilizados para transformar la información recibida y enviada hacia el servidor. Estos *beans* son creados gracias a las herramientas provistas en la librería ofrecida por fasterxml-jackson para la transformación de objetos java a JSON y viceversa. De este modo, la transformación se realiza de un modo simple y rápido. Estos modelos son manejados cada vez que se quiere realizar la inclusión, de nuevos elementos en el servidor, como pueden ser modelos, dispositivos o nuevas subscripciones de alertas, o si se quiere mostrar por pantalla la información acerca de un elemento almacenado en el servidor.

La interfaz gráfica se ha realizado utilizando la última versión hasta el momento de HTML, siendo HTML5, utilizándose por la creación de nuevos elementos que resultan más actuales y cómodos para desarrollar una interfaz web. Dejando atrás el uso de HTML versión 4, cuya utilización era más tosca y compleja. En el desarrollo se ha utilizado JSP para poder portar y controlar de una manera sencilla tanto el modelo como el controlador.

El estilo está gestionado a través de una hoja de estilos CSS que es asociada en la cabecera HTML de cada JSP. Mediante esta hoja de estilos se da forma a cada elemento mostrado en la interfaz web del componente. La hoja de estilos desarrollada es llamada "DashboardApplicationStyle.css".

Las animaciones y dinámicas que aparecen son implementadas mediante código JavaScript y jQuery. Estas animaciones son utilizadas por ejemplo para que de una manera sencilla puedan aparecer o desaparecer elementos y que se puedan ocultar a la vista del usuario elementos que realmente sí que están, de una manera dinámica sin tener que acudir a hojas de estilos.

Los diagramas mostrados son desarrollados con la librería basada en jQuery, jqPlot, que es utilizada para dibujar cualquier tipo de diagrama, ya sea de barras, de línea, o incluso diagramas de roscas. Estos diagramas que se muestran son utilizados para mostrar los datos medidos por los dispositivos y sus sensores.

El controlador se ha desarrollado utilizando el lenguaje de programación Java y utilizando las librerías ofrecidas por Spring para la creación de elementos según la arquitectura de software MVC, de modo que resulta más cómoda la creación de una aplicación siguiendo la estructura MVC. En la creación de este componente se han desarrollado 7 controladores, de los cuales 5 se corresponden con la interfaz gráfica mostrada en el portal web.

Desde los ficheros de configuración, el web.xml, un fichero de configuración en formato XML, se indica mediante la etiqueta "welcome-file" el fichero que se mostrará en el inicio en el momento de acceso a la web. El fichero que se indica es el llamado "index.jsp". En este fichero se describe la información mostrada en lenguaje HTML5 y se

describe el elemento "*iframe*" utilizado para cargar vistas dentro de otras vistas, es decir sobre este *iframe* se cargarán los modelos de cada controlador que disponga de interfaz gráfica a través del acceso al *path* correspondiente a cada uno de los controladores.

La distribución del contenido de utilizado en la interfaz gráfica es la siguiente:

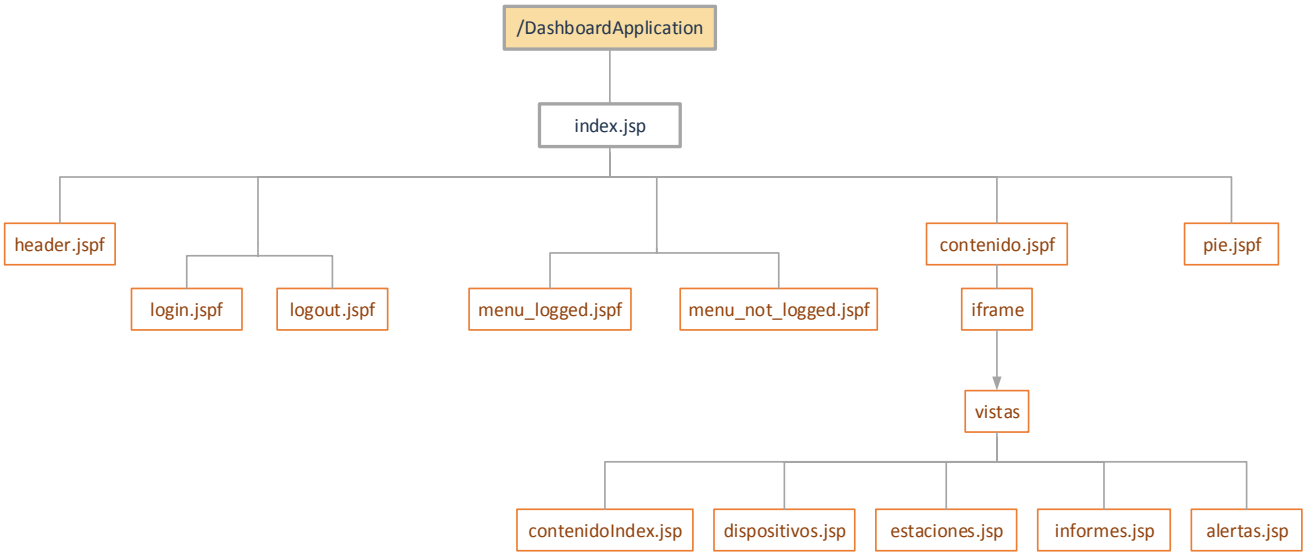

Figura 17. Distribución Dashboard Application

El contenido del fichero index.jsp está compuesto de JSPF. Estos ficheros son fragmentos de código JSP y pueden ser utilizados e incrustados en otros ficheros JSP. Por tanto, el fichero index.jsp, cuando es mostrado, muestra cinco fragmentos ya que los fragmentos de *login* o menú varían en función de si el usuario está autenticado o no, a partir del servicio de autenticación de M2MService y de la utilización de cookies. El fragmento JSP contenido.jspf, tiene el elemento *iframe* donde se cargarán todas las vistas, siendo inicialmente contenidoIndex.jsp, pero que seleccionando las opciones del menú mostradas a partir de menu_logged.jspf o menu_not_logged.jspf cargará las vistas en el contenedor *iframe*. Cada una de estas vistas está acompañada de su controlador.

A continuación se va a describir detalladamente cada funcionalidad desarrollada en el componente para que de una manera sencilla se pueda conocer como está establecida cada función y qué se ofrece con cada una de ellas.

3.5.1 GESTIÓN

Este controlador ha sido desarrollado para llevar toda la gestión de la información extraída del componente M2MService incluyendo su servicio de autenticación de usuarios.

En este caso no tiene un modelo concreto, ya que es accesible a través del resto de controladores. El *path* que se ha asociado para el acceso de las funciones de este controlador es "/DashboardApplication/gestion"

Las funciones que realiza son las siguientes:

AUTENTICACIÓN

La autenticación se compone de dos funciones: para el inicio de sesión y para la finalización de la sesión.

Mediante la función desarrollada para el inicio de sesión se establece comunicación con el servicio de autenticación de M2MService adjuntando el usuario y contraseña de quien se desee validar. Estos parámetros adjuntados son recogidos a partir de la web dentro de este cuadro:

Figura 18. Cuadro de Inicio de Sesión

En el proceso de inicio de sesión se registra una cookie con el valor del usuario que se ha registrado si la autenticación ha sido válida. A partir del control de esta cookie se gestiona la visualización de los elementos disponibles para todos y los elementos solo disponibles para el administrador del servicio.

El otro proceso de autenticación es la finalización de sesión, de este modo se invoca como en el proceso anterior a la funcionalidad del servicio de autenticación del componente. Para gestionar la visualización de los elementos correctamente se realiza el borrado de la cookie anteriormente almacenada para controlar los permisos de usuario.

Estos servicios se han desarrollado de un modo sencillo y realizan su función de una manera eficiente y sin excesiva complejidad, ya que en el desarrollo de sistemas de seguridad en páginas web existen multitud de implementaciones para la autenticación de usuarios desde implementaciones muy simples a implementaciones con alta carga de sistemas de seguridad como por ejemplo pueden ser sistemas encriptadores o sistemas como OAuth. En este caso no se requiere un alto grado de seguridad ya que su función es de demostrador.

AÑADIR DISPOSITIVO

La funcionalidad para añadir dispositivo se realiza para que de una manera correcta se realice el registro de nuevos dispositivos en la base de datos del DCA a través del componente M2MService. A partir de los datos recibidos desde el controlador de Dispositivos se realiza la llamada a esta función para que sea ella la que realice el registro.

Este registro se hace en el siguiente formulario:

Figura 19. Formulario Registro Dispositivo

El cliente que se crea se comunica con el recurso ofrecido por el componente con el *path* "/devices" y mediante una operación POST realiza la comunicación, enviando en el cuerpo del mensaje los datos del dispositivo a registrar en formato JSON como se especifica.

AÑADIR ALERTA

En esta funcionalidad se ofrece el registro de alertas a partir de la creación de reglas. Al igual que en la función anterior, esta es accesible a través de otro controlador, en este caso el controlador de Alertas, donde se muestra toda la información de las alertas creadas.

Se ofrece un formulario para añadir la información necesaria para crear un nuevo servicio de alertas para que posteriormente sean recibidos los avisos en tiempo real:

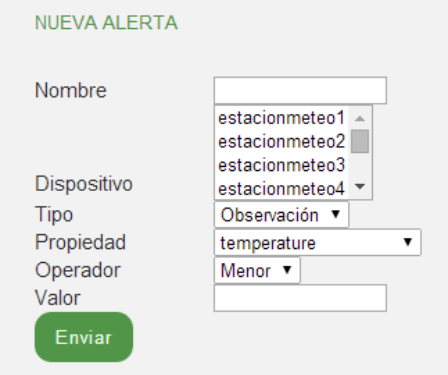

Figura 20. Formulario Registro Alerta

De igual modo que antes, el acceso al recurso ofrecido por el servicio REST es a través de una operación POST y se envía la información añadida a través de un JSON en el cuerpo del mensaje.

BORRAR DISPOSITIVO

El controlador Gestión ofrece funciones de borrado, tanto de dispositivo como de alertas, las cuales son accesibles a través de sus respectivas interfaces. En este caso, la función para borrar dispositivos accede al recurso cuyo *path* es "/devices" adjuntando el deviceId correspondiente con el dispositivo que se quiere borrar su registro, como se ha explicado anteriormente en la especificación del servicio REST de M2MService.

BORRAR ALERTA

En esta función, como se especifica en el anterior apartado, se realiza el borrado de una alerta de la base de datos. En este caso esta función realiza el acceso al recurso "/subscriptions" adjuntando el subscriptionId referido a una alerta.

GENERAR HISTÓRICO

Esta función es accedida a través del controlador Informes donde en su interfaz se realizan informes gráficos de las medidas realizadas por los dispositivos. Mediante esta función se ofrece la creación de un fichero XLSX de un histórico de todas las medidas mostradas en las gráficas.

Una vez mostrada una gráfica se muestra un botón donde pulsando el usuario recibirá la descarga del fichero.

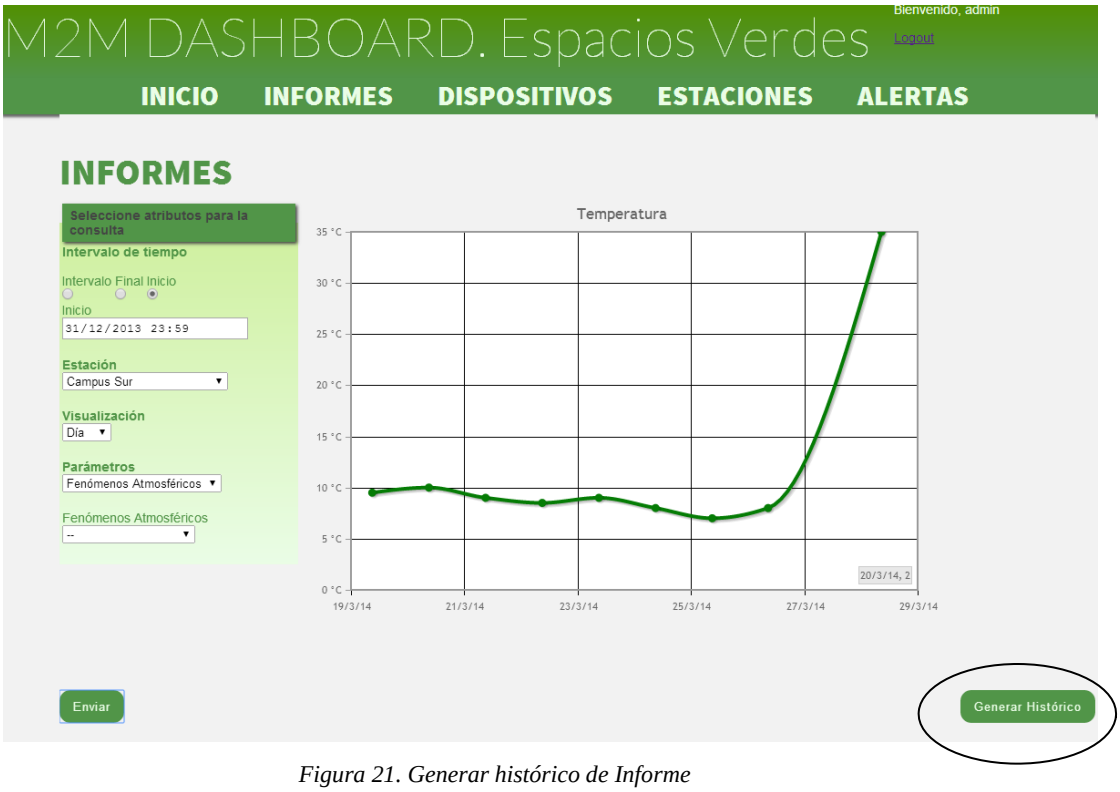

Figura 21. Generar histórico de Informe

El acceso a esta función es mediante un GET y es una funcionalidad exclusiva de Dashboard Application, no es ofrecida por el servicio REST y sirve de ejemplo de que soluciones se pueden desarrollar a partir de los servicios que ofrece Telefónica como el DCA.

GESTIÓN DATOS RECIBIDOS

La funcionalidad de gestión de los datos recibidos es referida a la transformación de la información recibida en bruto por el controlador. Cuando se ha mandado la acción desde la web el controlador Gestión realiza una petición GET al servicio REST de M2MService para que a partir de los parámetros de consulta indicados reciba un conjunto de datos que son las medidas registradas filtrada por los parámetros indicados en la consulta. Estos datos son transformados para que puedan ser leídos y mostrados en las gráficas realizadas mediante jqPlot, ya que el formato de los *arrays* de datos que lee debe de llevar un formato especial para su correcta visibilidad.

Por tanto desde el controlador Gestión se lleva a cabo toda la transformación de los datos mostrados en cada gráfica que aparece en la web. Desde la interfaz gráfica del resto de controladores se llama a estas funciones desarrolladas para la transformación del contenido.

3.5.2 INICIO

El controlador Inicio se encarga de cargar el modelo que utiliza el fichero "contenidoIndex.jsp". El *path* mediante el cual se accede a la interfaz gráfica de este controlador es "/DashboardApplication/doIndex", de modo que inicialmente es llamado desde el elemento *iframe* del fichero contenido.jspf que está incluido en index.jsp.

La interfaz gráfica ofrecida por este controlador se compone por cuatro bloques, siendo estos: Niveles de Gases, Consumo Agua Semanal, Última Hora y Recomendaciones.

Figura 22. Interfaz Inicio

El primer bloque, Niveles de Gases se ha desarrollado para mostrar información de las medidas realizadas en la calidad del aire en cada una de las estaciones. Mediante la utilización de dispositivos de control de la calidad del aire y control de gases se puede realizar la medida de distintos parámetros y componentes gaseosos, como por ejemplo son CO2, O2, NO2 o CH4 entre otros. Se ofrece la opción de selección de una estación base para poder mostrar información independiente de cada una de las estaciones. Los resultados mostrados corresponden a la última medida recibida de cada tipo de componente gaseoso.

El segundo bloque, quedando a la derecha de Niveles de Gases, es el bloque que muestra el consumo de agua realizado en una semana, mostrado como Consumo Agua Semanal. Mediante este bloque se muestra el consumo realizado en el riego de los jardines y demás plantación donde se encuentra una estación instalada. Se realiza control de gasto diario, mostrando en la gráfica la medida a lo largo de la última semana. Como anteriormente, también se dispone de la visualización de cada una de las estaciones asociadas con el servicio. Como trabajo futuro del proyecto realizado será la incorporación

de sistemas inteligentes que controlen el riego en función de las medidas realizadas en control de la humedad del terreno y de la cantidad de lluvia caída para que de este modo pueda observarse más claramente los efectos beneficiosos de estas soluciones en el consumo de los recursos naturales.

El tercer bloque es el reservado para Alertas. En este espacio se mostrarán las alertas recibidas en tiempo real desde el DCA en el componente M2MService. Desde Dashboard Application a través de su controlador Gestión se establece comunicación con el servicio REST de M2MService para recibir las alertas que se han producido y están almacenadas. Estas alertas deben haber sido previamente configuradas sus reglas para poder recibirlas, ya que cada alerta que se mostrará en este espacio es porque en alguna medida de un parámetro se ha superado, ha sido igual o inferior a un valor crítico establecido en las reglas. Las reglas han de ser configuradas en el espacio Alertas del Dashboard Application o accediendo directamente al servicio REST de M2MService desde un cliente REST habiéndose previamente autenticado como administrador del servicio.

El último bloque, Recomendaciones, no ha sido implementado siendo dejado para próximas evoluciones del servicio. Su función será el de mostrar posibles soluciones al administrador y a los propios usuarios que consulten la página para mejorar su eficiencia energética y disminuir el consumo de los recursos naturales como es el agua, ofreciendo ideas para que los usuarios puedan poner en práctica en sus propios jardines personales.

3.5.3 INFORMES

El controlador Informes se encarga de mostrar en la vista el fichero informes.jsp. Esta función es accesible para cualquier tipo de usuario, ya sea externo o administrador del servicio. La funcionalidad que ofrece es la capacidad de crear gráficas personalizadas para poder conocer las medidas realizadas en un lugar concreto, durante un periodo concreto y del parámetro que se quiera.

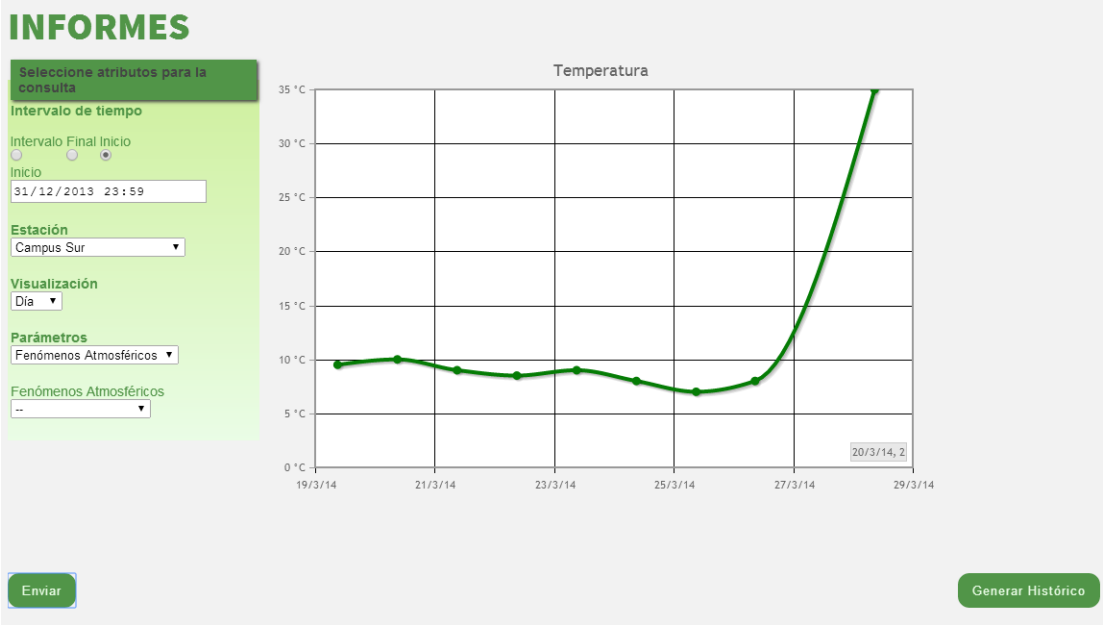

Figura 23. Interfaz Informes

En el formulario las opciones que se muestran son las siguientes:

- Fecha:

  Elección de unas fechas límites para un intervalo, asignar solamente un límite, o la no elección de límites temporales.

- Estación:

  En este apartado se ofrece la posibilidad de elegir entre las estaciones que se encuentren disponibles. Ya que aunque el parámetro final pueda ser el mismo, su valor medido podrá ser distinto ya que cada estación se encuentra en una localización diferente.

- Visualización:

  Se ofrecen 4 opciones de visualización, hora, día, mes y año. En función de la elección del tipo de visualización, se llevará acabo la media sobre los parámetros medidos en una misma hora, día, mes o año en función del valor seleccionado.

- Parámetros:

  En este apartado se debe seleccionar sobre qué parámetro se quiere mostrar la gráfica. Los parámetros disponibles se dividen en tres grupos, parámetros relacionados con los fenómenos atmosféricos, parámetros relacionados con la medida de partículas en el aire o por último, el riego.

Para nombrar otra opción que ofrece Informes, es la opción de generar un histórico, ya comentado en el anterior apartado del controlador de Gestión, donde se puede generar

un fichero XLSX donde se pueda saber de una manera detallada en una tabla los valores que se muestran en la gráfica.

3.5.4 DISPOSITIVOS

Dispositivos es el controlador asignado para la gestión de los dispositivos registrados en el servidor del DCA. Mediante la vista de este controlador se muestra una tabla los dispositivos que se encuentran actualmente registrados en la base de datos del DCA.

Cabe destacar que la vista de Dispositivos es únicamente accesible para el administrador, es decir, para poder acceder primero es obligatorio haber accedido con una cuenta válida, que el componente M2MService valide, de este modo se mostrará en la barra de menú la opción Dispositivos a través de la cual se accederá a la vista. Como se puede observar en la siguiente captura, el menú difiere de anteriores previsualizaciones ya que ahora muestra las dos opciones omitidas a personas ajena a la administración, siendo Dispositivos y Alertas.

Figura 24. Interfaz Dispositivos

Los valores que se muestra en la tabla son los que se han encontrado como más importantes y destacables para mostrar al usuario dentro de toda la información que rodea al registro de un dispositivo, y son los siguientes:

- Nombre:

  Este es el nombre e identificador con el que se ha registrado un dispositivo.

- Descripción:

Aquí se encuentra una breve descripción sobre el dispositivo mostrado.

- Modelo:

En este campo se identifica el modelo con el que se ha registrado cada dispositivo. El identificador que se muestra es único de cada modelo que está definido en el DCA.

- Localización:

La localización donde se encuentra el dispositivo. Este campo es un enlace para que posteriormente se pueda mostrar en un mapa mediante un marcador el lugar donde se encuentra instalado.

- Estado:

Aquí se muestra el estado del dispositivo, si se encuentra activo o no. Esto indica si se encuentra funcionando actualmente.

En la interfaz se muestran distintos botones, siendo estos las distintas opciones que se ofrecen. Como primera opción se muestra el botón Ver, seleccionable una vez que se haya seleccionado qué dispositivo ver en detalle los parámetros asociados con un dispositivo.

Figura 25. Detalle dispositivo

En el despliegue de esta vista se muestra la opción de Ver Último Registro para conocer las últimas medidas registradas por el dispositivo.

La opción Nuevo, muestra un nuevo formulario para añadir un nuevo dispositivo.

Figura 26. Formulario Nuevo Dispositivo

La opción Borrar se utiliza, como su nombre indica para eliminar la información asociada a un dispositivo ya registrado, primero se marca el dispositivo que se quiere eliminar, y posteriormente se pulsa el botón de Borrar.

Tanto en este apartado como en el resto cabe destacar que la comunicación con M2MService o para llamar servicios de otros controladores se utilizan las funciones de la librería jQuery. Mediante estas funciones para AJAX se realiza el envío de peticiones HTTP del tipo GET, POST y DELETE. Además se utiliza el mecanismo CORS [29] para establecer el acceso a orígenes distintos al servidor de destino. Este mecanismo viene definido en al documentación que proporciona W3C [30] para su correcto uso, ya que el desarrollo de este mecanismo es complejo. Cabe destacar la especial dificultad de la configuración a recursos mediante un POST o DELETE debido a la criticidad de estos servicios y la desestabilización que pueden provocar en los datos almacenados en el servidor.

Las opciones de Activar y Desactivar se muestran pero actualmente no está implementada la lógica de cada una.

Por último, otro elemento destacable de la vista, es el mapa mostrado. Este mapa mostrado es desarrollado a partir de la API de Google Maps V3 para poder desarrollar mapas personalizados. En el mapa se mostrará la localización de cada dispositivo una vez se pulse sobre el enlace asociado a su localización.

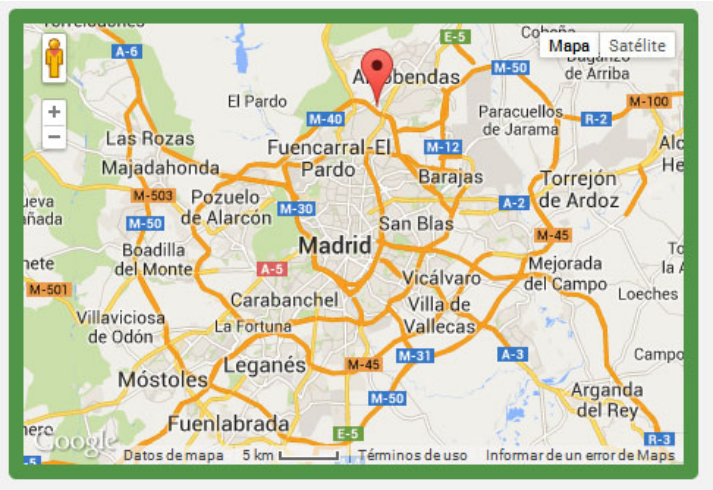

Figura 27. Mapa Dispositivos

3.5.5 ESTACIONES

Mediante el controlador Estaciones se realiza la muestra de los dispositivos instalados en las distintas estaciones que se han tomado como fijas. De modo que al igual que la anterior vista Dispositivos, se hace uso de la API ofrecida por Google Maps V3 [29] para mostrar un mapa personalizado. Es posible el uso de mapas de Google gracias a la creación de una cuenta de desarrollador donde se provee una clave que es necesaria para el posterior acceso a la API. En el mapa, se muestran las estaciones disponibles, que si son seleccionadas se mostrará los dispositivos que están instalados en cada una de ellas. En este caso, esta funcionalidad de conocer las medidas realizadas en cada estación es ofrecida para todos los usuarios, ya sean administrador o no del servicio.

Figura 28. Interfaz Estaciones

En la imagen del mapa se puede observar distintos marcadores que se muestran, cada uno de ellos corresponde con un conjunto de dispositivos, formando en sí lo que se ha definido con el nombre de Estación y es referido a un conjunto de dispositivos que estarán instalados en una misma localización centrada en unos jardines, parques o cualquier zona verde de la ciudad. Sobre cada marcador aparecen los dispositivos localizados en ese punto.

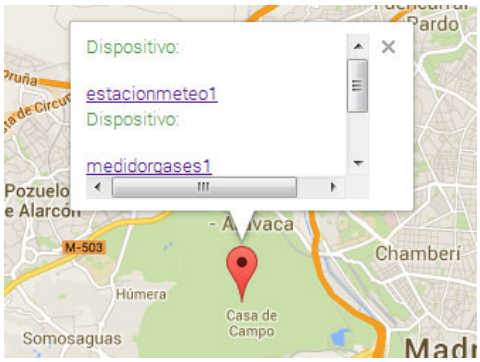

Figura 29. Marcador Estación

Una vez que se selecciona un dispositivo, se muestra a la derecha del mapa una tabla con los valores de las últimas medidas registradas en ese dispositivo:

Figura 30. Último Registro dispositivo

3.5.6 ALERTAS

Alertas es el último controlador y vista desarrollados dentro del componente Dashboard Application. Mediante este controlador se lleva a cabo la gestión de las reglas creadas para las alertas además de la posibilidad de borrar o crear nuevas reglas.

La vista asociada a este controlador solo es accesible si se ha autenticado previamente con permisos de administrador.

En la vista se ofrece una tabla donde se muestran las reglas creadas además de la posibilidad de poder crear nuevas reglas o borrar alguna de las ya existentes.

Figura 31. Interfaz Alertas

La tabla muestra los valores más identificativos de las reglas creadas siendo los siguientes:

- Nombre:

El nombre con el que se ha creado la regla para producir alertas.

- Identificador:

Código que identifica el tipo de alerta.

- Filtro:

Se muestra la regla configurada.

- Tipo:

Indica el tipo de regla creada, puede ser del tipo Observación o Registro

- Propiedad:

Indica sobre que propiedad de medida se está haciendo un control.

- Fecha de Creación:

Se muestra la fecha en la que se activó el registro de la regla para generar alertas.

4. RESULTADOS

En este apartado se van a realizar las pruebas necesarias para demostrar las capacidades que ofrece el demostrador realizado como solución en este proyecto. Primero se detalla que pruebas se van a realizar para dar una idea del funcionamiento de la plataforma.

Primero se va a mostrar cómo queda inicialmente el servicio recién desplegado, sin contener ningún tipo de registro. A continuación se mostrará el contenido de cada uno de los apartados, probando el sistema de autenticación para administradores del servicio, y mostrando que ofrece cada uno de los apartados.

Una vez detallado el contenido de los apartados mostrados, se comenzará a probar el registro de nuevos dispositivos. Esto se realizará una vez registrados los modelos que deberán seguir estos dispositivos, esta acción no es realizada en el demostrador sino que directamente en la plataforma que ofrece el DCA, siendo parte del posible trabajo futuro a partir de lo realizado para el demostrador.

A partir de que se haya realizado el registro de los nuevos dispositivos, se accederá al apartado de Estaciones para ver en el mapa mostrado, la localización de cada uno de los dispositivos. Además desde el propio apartado de Dispositivos donde se crea el registro de los dispositivos, se mostrará en detalle el contenido de alguno de ellos.

Después del registro de dispositivos, se quiere demostrar el funcionamiento de las alertas, por lo que se realizará el registro de nuevas reglas para poder provocar una alerta.

A partir de tener registrados ya dispositivos y reglas para controlar las posibles alertas en la monitorización de las medidas se llevará a cabo a partir del Simulador de Sensores proporcionado por Telefónica I+D para que así se pueda simular las medidas realizadas por los dispositivos como si fueran reales.

Finalmente se mostrará que tipo de control sobre las medidas realizadas se puede llevar a cabo gracias al apartado de Informes donde se puede realizar el control de las medidas realizadas por cada uno de los dispositivos.

4.1 PRIMEROS PASOS

Antes de comenzar a registrar dispositivos o crear las medidas realizadas por cada uno de los dispositivos, se debe de conocer que ofrece la solución propuesta en este proyecto, por tanto se mostrará cómo se ve cada uno de los apartados desarrollados. Primero como se comienza desde cero, el contenido de cada uno de los apartados se verá

que está vacío, pero a medida de que se vaya realizando las pruebas oportunas se mostrará contenido sobre cada uno de los apartados.

4.1.1 INICIO

El contenido de este apartado es muy simple, se ha separado en cuatro apartados, Nivel de Gases, Consumo de Agua Semanal, Alertas y Recomendaciones. La primera muestra el último registro realizado por los sensores del estado de la calidad del aire, mostrando parámetros como el nivel de Oxígeno, de Dióxido de Carbono entre otros. El segundo muestra el consumo realizado durante la semana del riego dentro de los jardines, mostrando las variaciones en función de si ha llovido, de si el terreno está demasiado húmedo como para realizar el riego o del periodo del año donde nos encontremos, ya que variará si es invierno o verano por ejemplo. En el apartado de Alertas se muestra las alertas que han saltado si se ha sobrepasado o alcanzado un nivel configurado previamente en las reglas creadas para las alertas. Por último, el apartado de Recomendaciones muestra como su nombre indica recomendaciones para la mejora en la eficiencia energética, el consumo de agua o de cómo mejorar la contaminación producida, este apartado se toma como trabajo futuro.

4.1.2 DISPOSITIVOS

En este apartado se muestra una lista de los dispositivos creados y diversas opciones para llevar a cabo el registro de nuevos dispositivos, borrar dispositivos ya registrados, ver en detalle la información relacionada con un dispositivo y otras opciones como Activar o Desactivar, dejadas como trabajo futuro, pensadas para no tener que borrar el registro de un dispositivo si se deja de utilizar temporalmente.

4.1.3 INFORMES

En este apartado se puede realizar informes gráficos del histórico de medidas realizadas por un dispositivo, en el momento de la captura, no se encuentra ningún dispositivo registrado y por tanto ninguna medida realizada, por lo que más adelante, se mostrará las funcionalidades que ofrece.

4.1.4 ESTACIONES

En el apartado de Estaciones se muestra geográfica la localización de cada uno de los dispositivos instalados. En este caso, de momento no se muestra ninguno disponible, pero como el caso anterior, cuando se haya realizado el registro de alguno de los dispositivos, se mostrará en el mapa su localización.

4.1.5 ALERTAS

Sobre este apartado se mostrará las reglas para la generación de alertas en función de las medidas realizadas por los dispositivos. Se muestra una lista de las reglas, y distintas opciones a continuación, la posibilidad de creación de nuevas reglas, o el borrado, inclusive el activar o desactivar éstas, siendo esto preparado como trabajo futuro a partir de la solución creada.

4.2 REGISTRO DE DISPOSITIVOS

Para el proceso de registro de dispositivos, inicialmente debe de crearse modelos de dispositivos, donde se va a indicar que tipo de parámetros y sensores va a contener cada dispositivo. En este caso, al tratarse de una simulación, la creación de un modelo basta para que posteriormente se registre un dispositivo y a partir de ahí ya se puedan crear medidas simuladas por ese dispositivo.

La creación de los modelos no está disponible en el servicio creado, sino que es necesario crear los modelos a partir de la herramienta ofrecida por DCA para la gestión del contenido del usuario en sus bases de datos. Además es necesario llevar a cabo la localización de los espacios verdes donde se van a instalar los dispositivos, para esta demostración se ha optado por la selección de tres localizaciones en la ciudad de Valladolid, las cuales son:

- La Rosaleda (41.6519137,-4.7332118). Identificador 1
- Plaza de Juan de Austria (41.638823,-4.742078). Identificador 2
- Parque de la Paz (41.6346014,-4.7260144). Identificador 3

Para realizar el registro en otras localizaciones debería modificarse sus referencias en Dashboard Application.

Para la creación de dispositivos, se utiliza el formulario creado en la aplicación para mostrar su utilidad como se ve en la figura 19 mostrada anteriormente.

Los dispositivos que se pretenden simular son de tres tipos: estación meteorológica, medidor de partículas y controlador de riego. El primero se encarga de controlar

parámetros atmosféricos, el segundo la calidad del aire, y el tercero el flujo de agua. La lista de dispositivos queda de la siguiente manera como se muestra en la figura número 32.

Figura 32. Lista de Dispositivos Registrados

4.3 REGISTRO DE ALERTAS

Las alertas son creadas para el control de valores límite, estos son establecidos mediante reglas y Dashboard Application exporta la utilizad ofrecida por DCA para la creación de las reglas. De este modo y de un modo similar que con el registro de dispositivos, en la pestaña de alertas se puede crear nuevas reglas a través de un formulario. En este formulario se elige los dispositivos sobre los que se quiere aplicar la regla y el parámetro a controlar. En la figura 20 se muestra el formulario que se debe de rellenar para crear una regla de alertas.

Una vez realizada la creación de todas las reglas se mostrarán en una lista como se puede observar en la figura 33. En este caso se han realizado por el momento dos reglas.

Figura 33. Lista de Reglas de Alertas Registradas

Para realizar la eliminación de una regla se debe seleccionar la deseada y proceder al borrado pulsando sobre Borrar.

Como ejemplo se crea un registro de un valor de temperatura mayor a 35 grados, tal y como está establecida la alerta y si se accede al inicio del portal podemos observar que se ha registrado la alerta instantáneamente como se ve en la figura siguiente.

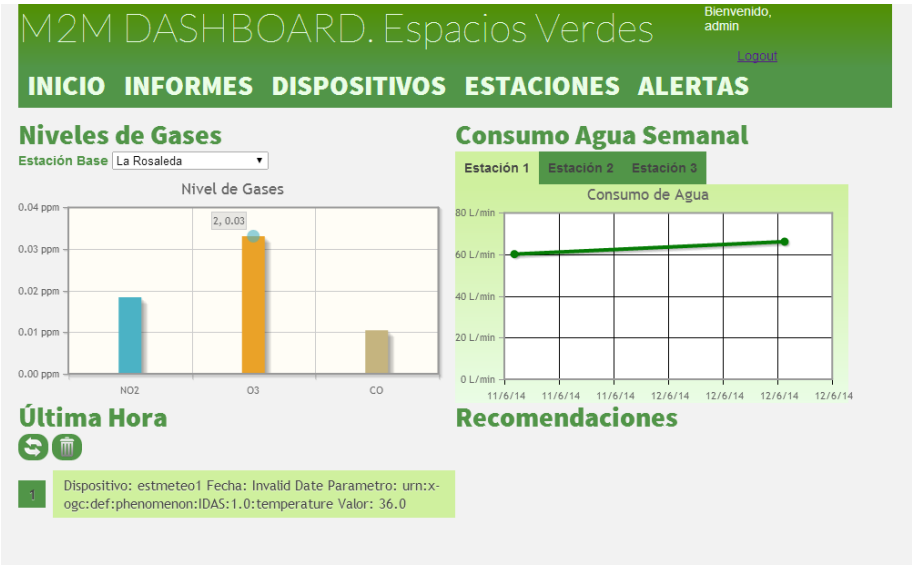

Figura 34. Creación Alerta

A su vez se ha enviado un SMS al administrador del servicio para que conozca qué tipo de alerta se ha registrado. Como trabajo futuro estas alertas podrían ser enviadas a unos clientes con mayor acceso y mediante un registro de usuarios donde se indique el número de teléfono móvil donde se desee recibir las alertas.

4.4 Creación de Contenido

Ahora se procede a crear información simulada de cada uno de los dispositivos registrados en el sistema. Para ello se hace uso de la extensión de Google Chrome, Advanced Rest Client Application. Para enviar mediante peticiones HTTP al servicio REST de DCA para el registro de las medidas realizadas.

Un ejemplo de cómo sería la adición de contenido se puede ver en la siguiente figura.

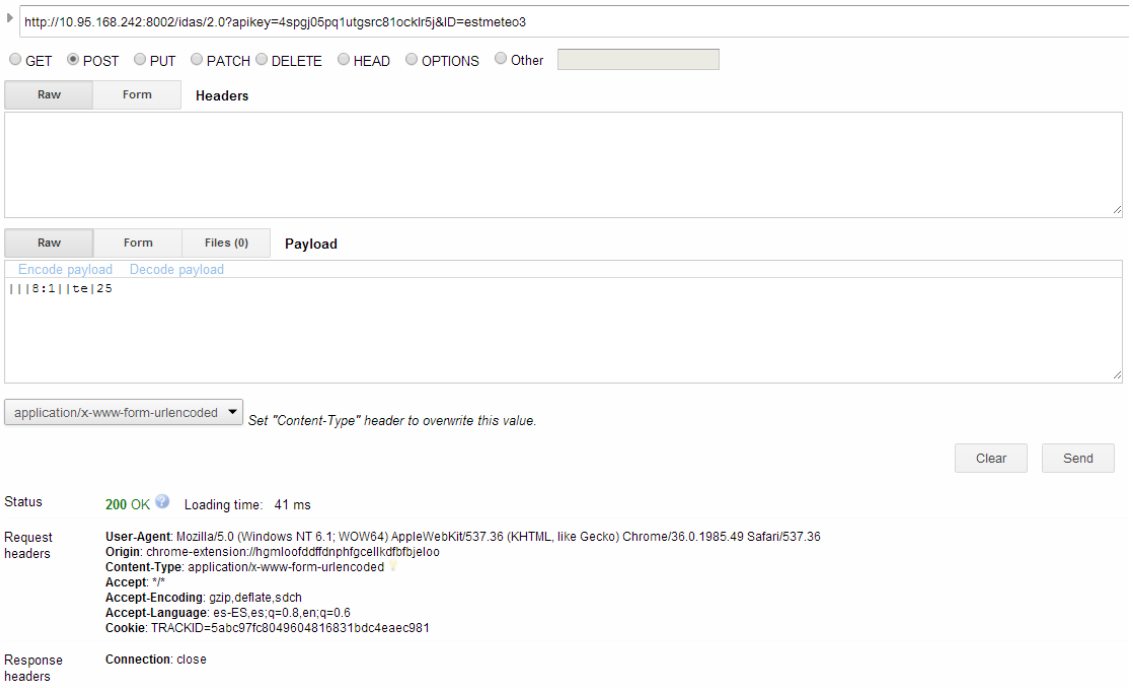

Figura 35. Advanced Rest Client Application

La regla utilizada para crear contenido es la siguiente:

- *Endpoint:* {host}?apikey=CLAVEDCA&ID=deviceID
- *Body:*|||CODIGO DEL PARÁMETRO||ALIAS DEL PARÁMETRO|VALOR

De este modo se ha creado información y medidas ajustadas a los posibles valores reales que pueden medirse en las localizaciones indicadas anteriormente. Como se ve en la imagen, como respuesta al envío de la medida, si se ha realizado correctamente se contesta con un mensaje con estado 200 OK.

Para comprobar que se ha registrado información, se puede acceder directamente al recurso ofrecido por M2MService, o a través de la interfaz de Dashboard Application. Esta última accede a todo el registro de medidas en el apartado Informes donde en función de la configuración y filtros especificados se accederá a toda o a una parte de la información. Como ejemplo se muestra en la figura siguiente cómo sería el JSON que se recibe si se quiere comprobar los datos registrados por un dispositivo y sus sensores.

```
{
    "count": 1,
    "asset": "estmeteo3",
    "data": [
        {
            "st": "2014-06-11T11:44:49Z",
            "ms": {
                "p": "temperature",
                "v": "25.0",
                "u": "celsius"
            },
            "pms": [
                {
                    "p": "QoS",
                    "v": "0",
                    "u": ""
                }
            ]
        }
    ]
}
```


Figura 36. Registro de Medidas de Dispositivo

El registro de medidas se realiza a partir de unos identificadores descritos en los manuales del DCA [28], y los que se han usado para este demostrador han sido los siguientes:

Fenómeno	Phenomenon ID	Ultralight Name
Fenómenos atmosféricos	temperature	8:1
	atmosphericPressure	8:9
	rainfall	8:10
	windSpeed	8:7
	windDirection	8:5
	relativeHumidity	8:3
Medida de Partículas	NO2Concentration	8:71
	NOConcentration	8:13
	O2Concentration	8:70
	O3Concentration	8:14
	CH4Concentration	8:69
	CO2Concentration	8:15
	COConcentration	8:16
	gasConcentration	8:50
Riego	waterFlow	8:67
Luminosidad	luminance	8:51

Se ha optado por realizar distintos registros con una larga duración de tiempo, para poder mostrar las capacidades de la creación de informes que se verá en el apartado siguiente, además de las funcionalidades ofrecidas en el apartado de Inicio donde se muestra el último registro de la calidad del aire de cada una de las estaciones y el gasto realizado de agua en la última semana.

Primero, se ha realizado un registro del flujo de litros de agua por minuto que se podría haber dado en cada una de las estaciones registradas. El registro se muestra en el

apartado Índice, en Consumo de Agua Semanal, donde se reflejará el consumo de agua, y la intensidad con la que se ha realizado el riego en cada una de las estaciones. De modo que en futuras implementaciones este consumo baje en función del resto de niveles de humedad o precipitación de manera inteligente.

A continuación se realiza una prueba de registro de medidas de niveles de la calidad del aire a partir de documentación oficial de Agencia Meteorológica de Valladolid [30].

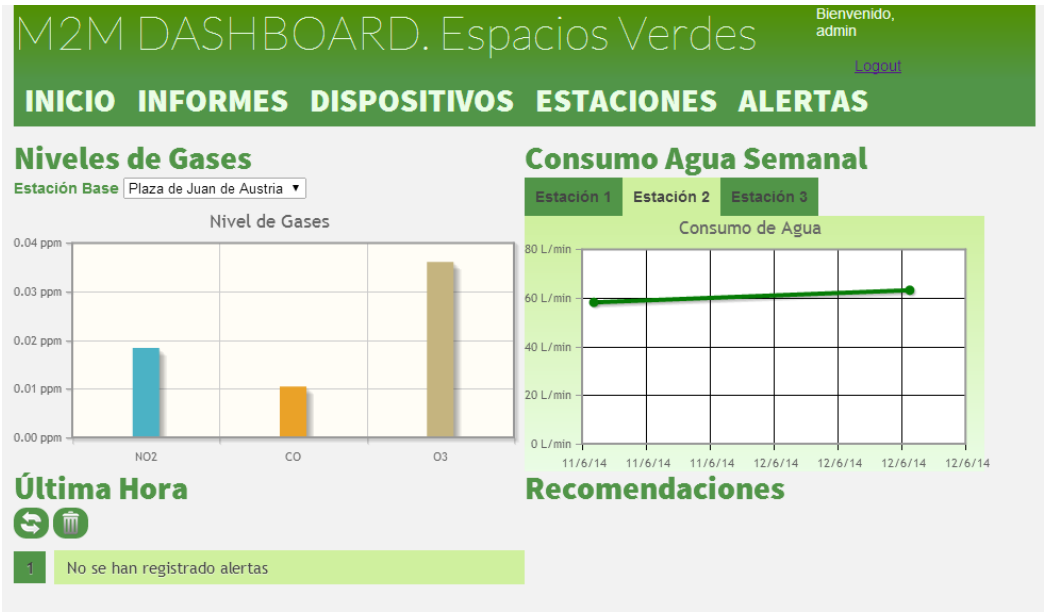

Figura 37. Datos Mostrados en Inicio

Se muestra como se mostraría los datos registrados, en Nivel de Gases se muestra la última medida realizada. En este caso se muestran los contaminantes NO2, CO y O3 pero además se puede mostrar niveles de CO2 u O2 entre otros.

Como prueba del registro de registros relacionados con fenómenos atmosféricos se realiza el registro de un histórico de temperaturas y niveles de humedad en cada una de las estaciones para la posterior muestra en informes y gráficas.

4.5 Creación de Informes

Una vez que se ha creado suficiente información de las medidas realizadas, se procede a mostrar una de las partes más complejas de Dashboard Application, que es su creador de gráficas e informes.

Como ejemplo de un informe y una gráfica es la siguiente figura que muestra una serie de valores de temperatura que se han registrado manualmente en DCA. Las gráficas ofrecen la posibilidad de realizar una consulta durante un periodo concreto o a partir de una fecha o con una fecha final. Del mismo modo también se permite realizar una visualización por hora, día, mes y año, realizando cada vez la media de cada uno de los

valores anteriores. Además permite mostrar una gráfica de todos los parámetros que sean medidos y haya valores registrados en las bases de datos del DCA.

Figura 38. Gráfica de Temperatura

Otra opción de muestra de la información es a través de un Excel creado con la información recibida a través de M2MService, utilizada para crear la gráfica. Este documento de Excel se puede descargar pulsando sobre Generar Histórico, y un ejemplo de ello es la siguiente captura realizada.

	A	B
1	FECHA	VALOR
2	06/01/201	15
3	06/02/201	16.5
4	06/03/201	17
5	06/04/201	18
6	06/05/201	22.6
7	06/06/201	21
8	06/07/201	17
9	06/08/201	21
10	06/09/201	22
11	06/10/201	22
12	06/11/201	23.5
13	06/11/201	25.5
14	06/12/201	19
15	06/13/201	23.5

Figura 39. Histórico

Otro ejemplo de creación de datos es el de humedad como se ha comentado anteriormente, mostrando las unidades de porcentaje.

Figura 40. Gráfica de Humedad

Ejemplos realizados para consulta utilizando intervalos es el siguiente *endpoint*:

```
http://10.95.14.162:8080/DashboardApplication/gestion/informes/3/estmeteo/relativ
eHumidity/2014-06-01T00:00:00Z%3C%3E2014-06-19T00:00:00Z/
```

De este modo se lanza una petición al *controller* Gestión de Dashboard Application que se encarga de realizar la consulta a M2MService, este se encarga de la comunicación fiable con DCA, y finalmente recuperar los datos.

5. CONCLUSIONES Y TRABAJOS FUTUROS

La solución se ha creado con la finalidad de servir como proyecto fin de grado del alumno, siendo ésta una muestra de las capacidades y conocimientos aquiridos por él durante sus años de estudios universitarios y durante el transcurso de la beca realizada en Telefónica I+D desde mayo de 2013.

El trabajo realizado ha sido desarrollado a partir de conocimientos previos como son Java o Javascript y del aprendizaje de nuevos como son librerías Java y utilidades como Spring *framework*, herramientas de gestión de proyectos en Java como Maven y de desarrollo web, tanto en *front-end* (HTML5, CSS3, jQuery, jqPlot) como en *back-end* (Spring mvc, Tomcat, REST).

El demostrador, realizado asociado a un proyecto de investigación de Telefónica I+D, tiene como núcleo la plataforma DCA de Telefónica I+D, que permite el desarrollo de aplicaciones basadas en redes de sensores de modo que a partir de distintas tecnologías se pueda desarrollar distintas aplicaciones enmarcadas en un entorno de M2M. El demostrador permite realizar una prueba de viabilidad del modo en el que se pueden desarrollar nuevos servicios, aplicaciones e ideas de negocio a partir de las capacidades e *enablers* ofrecidos y desarrollados por Telefónica. Todos los servicios son exportados utilizando la plataforma neoSDP para llevar el control del acceso a los *enablers* y los clientes de Telefónica.

La planificación previa mostrada en el Anteproyecto, se ha ido cumpliendo en cuanto a las fases aunque los plazos han ido variando en función de la dificultad de cada uno de ellas. Los dos componentes desarrollados han sido probados de manera exitosa simulando un caso real, desde el punto de vista de un usuario y desde la vista del administrador del servicio. La creación del servicio REST del componente M2MService ha tenido una especial dificultad en la transformación y entendimiento de los datos enviados y recibidos tanto en la comunicación con los usuarios como en su comunicación con DCA. Los problemas principales surgidos han estado relacionados con el tipo de contenido del mensaje y la sintaxis de éste para que concordara con los *beans* realizados en Java.

El desarrollo del componente Dashboard Application ha resultado más complejo ya que las tecnologías utilizadas (HTML5, CSS3, jQuery o las API ofrecidas por Google para sus servicios de mapas) eran de reciente adquisición. La comunicación con el componente M2MService se ha realizado mediante peticiones HTTP utilizando jQuery y sus funciones específicas para AJAX. La comunicación se realizaba con el servicio REST de M2MService mediante métodos POST, GET y DELETE. Se hace uso del mecanismo CORS, utilizado para definir los orígenes que pueden acceder a recursos de otros dominios.

Para la creación de los informes gráficos se ha hecho uso de la librería jqPlot, desarrollada a partir de jQuery y que al ser de libre distribución se puede utilizar sin coste. Ofrece un servicio muy útil para la creación de información gráfica a partir de un histórico de datos de un modo muy personalizable por parte del desarrollador. De este modo, en el apartado Informes de Dashboard Application, se permite que el usuario pueda seleccionar que tipo de gráfica quiere visualizar fijando el tipo de contenido, las unidades temporales y el rango de fechas de los datos que quiere obtener.

En definitiva, a partir de la creación de estos componentes, se ha presentado un demostrador que hace uso de tecnologías M2M para la creación de nuevos servicios en el entorno de *Smart Environment*. Este demostrador tiene la finalidad de ser uno de los muchos puntos de partida para que empresas creen nuevos servicios tecnológicos que mejoren la calidad de vida de las personas, mejoren el uso energético realizado y además reduzcan el daño continuo realizado sobre el medio que nos rodea.

Todo el trabajo aquí mostrado ha sido realizado por el estudiante, autor del mismo, gracias en parte a la ayuda de sus compañeros de trabajo durante la beca realizada, que han servido de otra fuente de consulta más..

5.1 LÍNEAS DE TRABAJO FUTURO

El demostrador no se ha realizado con la intención de que tenga un final cerrado, sino que sea escalable y se permita nuevos desarrollos a partir de éste y además de que sirva de ejemplo para nuevos servicios utilizando al menos uno de los componentes mostrados en esta memoria como son Dashboard Application o M2MService.

Como desarrollo futuro sobre el demostrador, se plantea la posibilidad, primero, de que haga uso de sensores reales, no siendo estos sensores simulados con herramientas de Telefónica I+D como resulta en este proyecto.

Segundo, llevar el control de actuadores y sensores en los que no sólo se monitorice un parámetro y se realicen informes gráficos, sino que, a través del portal web realizado en Dashboard Application, se puedan llevar a cabo diversas funciones sobre estos. Ejemplo de estas nuevas funciones serían el control del riego inteligente, es decir, que a partir de algoritmos en función de la lluvia recogida, y en función de la humedad de la tierra se varíe la cantidad de agua utilizada para el riego y el tiempo que está funcionando. De este modo se conseguirá una reducción considerable del uso de los recursos.

Tercero, se plantea la posibilidad de añadir nuevas funcionalidades en la creación y registro de dispositivos y alertas en el portal, mostrando todas las capacidades ofrecidas por DCA, no solo las consideradas fundamentales y básicas para el componente M2MService. Otro objeto posible de ampliación es el servicio de autenticación para usuarios que no solo sean administradores y así que exista también la posibilidad de recibir alertas vía SMS de las alertas producidas en el sistema.

Cuarto, otro objetivo para un futuro desarrollo es que exista la posibilidad de acceder a Dashboard Application a través de distintos terminales, es decir que sea

multiplataforma. A partir de aplicaciones móviles (iOs, Android, etc) o de sobremesa se permita el acceso a todo el conjunto de funcionalidades ya ofrecidas.

Por último y como objetivo principal de este demostrador, es que terceros, externos a Telefónica, contraten los servicios ofrecidos por Telefónica, ya que ofrecen un servicio estable y consolidado de modo que sólo sea necesaria la búsqueda de ideas dentro de este sector de las nuevas tecnologías y la *Smart City*. Ya que gracias a los recursos que se ofrecen, el desarrollo de nuevos servicios es más sencillo y su desarrollo no comenzará de cero ya que la infraestructura necesaria ya está creada, reduciendo el *time-to-market* de estos servicios.

6. BIBLIOGRAFÍA

[1] neoSDP Team, "neoSDP Architecture Components," 2012.

[2] DCA Team, "QuickGuide 2.2," 2013.

[3] Telefónica, "Smart City Telefónica," [Online]. Available: http://smartcity-telefonica.com/.

[4] G. d. Cataluña, "transìt," [Online]. Available: http://www20.gencat.cat/portal/site/transit. [Accessed 2014].

[5] SmartSantander, "SmartSantander – WP4, D4.2 Description of implemented IoT services," 2012.

[6] S. F. Programme, "Seven Framework Programme," [Online]. Available: http://ec.europa.eu/research/fp7/index_en.cfm.

[7] Telefónica, "Energy Manager," [Online]. Available: http://powerhousemanager.tid.es/EnergyManager/home.html.

[8] Telefónica, "PowerHouse," [Online]. Available: http://powerhousemanager.tid.es/WhiteLabel/auth/WhiteLabel.do.

[9] 7. F. Programme, "OUTSMART," [Online]. Available: http://fi-ppp-outsmart.eu/en-uk/Pages/default.aspx.

[10] A. M. Sugeeswari Lekamge, "Developing a Smart City Model that Ensures the Optimum Utilization," Japón, 2013.

[11] T. E. Foundation. [Online]. Available: https://www.eclipse.org/.

[12] A. S. Foundation, "Apache Tomcat," [Online]. Available: http://tomcat.apache.org/.

[13] HTML5, "HTML5 Rocks," [Online]. Available: http://www.html5rocks.com/en/.

[14] w3schools, "CSS," [Online]. Available: http://www.w3schools.com/css.

[15] J. Eguiluz, "Librosweb Javascript," [Online]. Available: http://librosweb.es/javascript/.

[16] jQuery, "jQuery," [Online]. Available: http://jquery.com/.

[17] jqPlot, "jqPlot," [Online]. Available: http://www.jqplot.com/.

[18] Wikipedia. [Online]. Available: http://es.wikipedia.org/wiki/JavaServer_Pages.

[19] Wikipedia. [Online]. Available: http://es.wikipedia.org/wiki/Java_(lenguaje_de_programaci%C3%B3n).

[20] Spring, "Spring Security," [Online]. Available: http://projects.spring.io/spring-security/.

[21] Spring. [Online]. Available: http://projects.spring.io/spring-ws/.

[22] R. N. Marset, "Modelado, Diseño e Implementación de Servicios Web".

[23] Wikipedia. [Online]. Available: http://es.wikipedia.org/wiki/JSON.

[24] FasterXML. [Online]. Available: http://wiki.fasterxml.com/JacksonHome.

[25] A. S. Foundation, "Apache Maven," [Online]. Available: http://maven.apache.org/.

[26] J. F. KJ (Ken) Salchow, "Load Balancing 101 Nuts and Bolts".

[27] BAELDUNG, "Java, Spring and Web Development tutorials," [Online]. Available: http://www.baeldung.com/2011/10/31/securing-a-restful-web-service-with-spring-security-3-1-part-3/.

[28] DCA Group, UNICA API M2M REST v2.2, 2013.

[29] M. Hossain, "Using CORS," [Online]. Available: http://www.html5rocks.com/en/tutorials/cors/?redirect_from_locale=es.

[30] W3C, "Cross-Origin Resource Sharing," 2014. [Online]. Available: http://www.w3.org/TR/cors/.

[31] Google, "Google Maps API v3," [Online]. Available: https://developers.google.com/maps/documentation/javascript/?hl=es.

[32] RCCAVA, "Biblioteca de la RCCAVA," [Online]. Available: http://www10.ava.es/rccava/documentos/bica/bica02.pdf.